On the vehicle routing problem with time windows

Kallehauge, Brian

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

On the vehicle routing problem with time windows

Brian Kallehauge

Submitted in partial fulfillment of the requirements
of the degree of Doctor of Philosophy

Centre for Traffic and Transport
Technical University of Denmark

January 2006

Abstract

The vehicle routing problem with time windows is concerned with the optimal routing of a fleet of vehicles between a depot and a number of customers that must be visited within a specified time interval, called a time window. The purpose of this thesis is to develop new and efficient solution techniques for solving the vehicle routing problem with time windows (VRPTW). The thesis consists of a section of introductory remarks and four independent papers.

The first paper 'Formulations and exact approaches for the vehicle routing problem with time windows' (Kallehauge, 2005, unpublished) is a review of the exact algorithms proposed in the last three decades for the solution of the vehicle routing problem with time windows. A detailed analysis of the formulations of the VRPTW is presented together with a review of the literature related to the different formulations. We present the two main lines of development in relation to the exact approaches for the VRPTW. One is concerned with the general decomposition approach and the solution to certain dual problems associated with the VRPTW. Another more recent direction is concerned with the analysis of the polyhedral structure of the VRPTW. We conclude by examining possible future lines of research in the area of the VRPTW.

In the second paper 'Lagrangian duality applied to the vehicle routing problem with time windows' (Kallehauge, Larsen, and Madsen, Computers & Operations Research, 33:1464-1487, 2006) we consider the Lagrangian relaxation of the constraint set requiring that each customer must be served by exactly one vehicle yielding a constrained shortest path subproblem. We present a stabilized cutting-plane algorithm within the framework of linear programming for solving the associated Lagrangian dual problem. This algorithm creates easier constrained shortest path subproblems because less negative cycles are introduced and it leads to faster multiplier convergence due to a stabilization of the dual variables. We have embedded the stabilized cutting-plane algorithm in a branch-and-bound search and introduce strong valid inequalities at the master problem level by Lagrangian relaxation. The result is a Lagrangian branch-and-cut-and-price (LBCP) algorithm for the VRPTW. Making use of this acceleration strategy at the master problem level gives a significant speed-up compared to algorithms in the literature based on traditional column generation. We have solved two test problems introduced in 2001 by Gehring and Homberger with 400 and 1000 customers respectively, which to date are the largest problems ever solved to optimality. We have implemented the LBCP algorithm using the ABACUS open-source framework for solving mixed-integer linear-programs by branch, cut, and price.

In the third paper 'Path inequalities for the vehicle routing problem with time windows' (Kallehauge, Boland, and Madsen, 2005, submitted) we introduce a new formulation of the VRPTW involving only binary variables associated with the arcs in the underlying digraph. The new formulation is based on a formulation of the asymmetric traveling salesman problem with time windows and has the advantage of avoiding additional variables and linking constraints. In the new formulation of the VRPTW time windows are modeled using path inequalities. The path inequalities eliminate time and capacity infeasible paths. We present a new class of strengthened path inequalities based on polyhedral results obtained in the context of the asymmetric traveling salesman problem with replenishment arcs. We study the VRPTW polytope and determine the polytope dimension. We show that the lifted path inequalities are facet defining under certain assumptions. We also introduce precedence constraints in the context of the VRPTW. Computational experiments are performed with a branch-and-cut algorithm on the Solomon test problems with wide time windows. Based on results on 25-node problems the outcome is that the algorithm shows promising results compared to leading algorithms in the literature. In particular we report a solution to a previously unsolved 50-node Solomon test problem R208. The conclusion is therefore that the path formulation of the VRPTW is no longer the unchallenged winning strategy for solving the VRPTW.

The fourth and final paper 'Vehicle routing problem with time windows' (Kallehauge, Larsen, Madsen, and Solomon. In Desaulniers, Desrosiers, and Solomon, editors, Column generation, pages 67-98, Springer, New York, 2005) is a contribution to a book on column generation edited by G. Desaulniers,

J. Desrosiers, and M. M. Solomon. The focus of the paper is on the VRPTW as one of the important applications of column generation in integer programming. We discuss the VRPTW in terms of its mathematical modeling, its structure and decomposition alternatives. We then present the master problem and the subproblem for the column generation approach, respectively. Next, we illustrate a branch-and-bound framework and address acceleration strategies used to increase the efficiency of branch-and-price methods. Then, we describe generalizations of the problem and report computational results for the classic Solomon test sets. Finally, we present our conclusions and discuss some open problems.

Resumé

Den danske titel på denne afhandling er 'Ruteplanlægningsproblemet med tidsvinduer'. Dette problem omhandler den optimale styring af en flåde af lastbiler mellem et lager og et antal kunder, der skal besøges inden for et bestemt tidsinterval, et såkaldt tidsvindue. Formålet med denne afhandling er udvikling af nye og effektive metoder til løsning af ruteplanlægningsproblemet med tidsvinduer (vehicle routing problem with time windows - VRPTW). Afhandlingen består af et afsnit af introducerende bemærkninger og fire separate artikler. Det introducerende afsnit beskriver artiklernes videnskabelige bidrag. I det følgende vil artiklernes engelske titel ikke blive oversat til dansk.

Den første artikel 'Formulations and exact approaches for the vehicle routing problem with time windows' (Kallehauge 2005, ikke publiceret) er en gennemgang af eksakte metoder udviklet gennem de seneste tre årtier til løsning af ruteplanlægningsproblemet med tidsvinduer. Der præsenteres en detaljeret analyse af formuleringerne af VRPTW, samt en gennemgang af den relevante litteratur for de forskellige formuleringer. De to hovedretninger i forskningen inden for eksakte metoder til VRPTW beskrives. En af retningerne omhandler generel dekomposition og løsning af visse duale problemer forbundet med VRPTW. En anden og nyere retning omhandler analyse af den polyhedrale struktur forbundet med VRPTW. Artiklen afsluttes med en diskussion af mulige fremtidige forskningsområder inden for VRPTW.

I den anden artikel 'Lagrangian duality applied to the vehicle routing problem with time windows' (Kallehauge, Larsen, and Madsen, Computers & Operations Research, 33:1464-1487, 2006) betragter vi en Lagrange relaksering af VRPTW. De restriktioner der kræver at hver kunde besøges af netop en lastbil relakseres, hvilket resulterer i et korteste vej subproblem med bibetingelser. Der præsenteres en stabiliseret snitplansalgoritme inden for rammerne af lineær programmering. Algoritmen anvendes til løsning af det Lagrange duale problem. Denne algoritme resulterer i nemmere korteste vej subproblemer, fordi færre negative kredse introduceres, og den resulterer i bedre konvergens af løsningen, fordi de duale variable stabiliseres. Vi har inkluderet den stabiliserede snitplansalgoritme i en branch-and-bound søgning og introducerer stærke gyldige uligheder i master problemet ved hjælp af Lagrange relaksering. Resultatet er en Lagrangian branch-and-cut-and-price (LBCP) algoritme til løsning af VRPTW. Benyttelsen af denne algoritme giver en signifikant forbedring af løsningstiden sammenlignet med algoritmer i litteraturen baseret på sædvanlig søjlegenerering. Vi har løst to test problemer introduceret i 2001 af Gehring og Homberger med henholdsvist 400 og 1000 kunder. Disse problemer er de til dato største problemer løst til optimalitet. Algoritmen er implementeret ved hjælp af open-source rammesystemet ABACUS, der er beregnet til løsning af heltalsproblemer ved hjælp af branch-and-cut-and-price.

I den tredje artikel 'Path inequalities for the vehicle routing problem with time windows' (Kallehauge, Boland, and Madsen, 2005, indsendt til publicering) præsenterer vi en ny formulering af VRPTW der kun involverer binære variable tilknyttet kanterne i en underliggende orienteret graf. Denne nye formulering er baseret på en formulering af traveling salesman problemet (TSP) med tidsvinduer og har den fordel at man undgår ekstra variable og koblende begrænsninger. I den nye formulering af VRPTW modelleres tidsvinduer ved hjælp af uligheder, der eliminerer ugyldige veje i netværket. En ugyldig vej kan skyldes overskridelse af kapacitet eller tidsbegrænsninger. Vi præsenterer en ny klasse af uligheder baseret på polyhedrale resultater opnået inden for TSP med replenishment begrænsninger. Vi bestemmer dimensionen af VRPTW polytopen. Vi beviser under visse antagelser at de nye uligheder er facetter for VRPTW polytopen. Vi introducerer også precedence begrænsinger for VRPTW. Beregningsmæssige eksperimenter udføres med en branch-and-cut algoritme. Vi betragter Solomons test problemer med brede tidsvinduer. Baseret på resultater for problemer med 25 kunder er konklusionen at algoritmen er lovende sammenlignet med førende algoritmer i litteraturen. Vi præsenterer også en løsning til et hidtil uløst problem med 50 kunder, nemlig R208. Konklusionen er derfor at korteste vej dekompositionen af VRPTW ikke længere er den absolutte vinderstrategi til løsning af VRPTW.

Den fjerde og sidste artikel 'Vehicle routing problem with time windows' (Kallehauge, Larsen, Madsen, and Solomon. In Desaulniers, Desrosiers, and Solomon, editors, Column generation, pages 67-98,

Springer, New York, 2005) er et bidrag til en bog om søjlegenerering redigeret af G. Desaulniers, J. Desrosiers og M. M. Solomon. Fokus for denne artikel er VRPTW som et vigtigt eksempel på anvendelse af søjlegenerering i heltalsprogrammering. Vi diskuterer VRPTW i henhold til modelleringsmæssige aspekter, problemstrukturen og dekompositionsalternativer. Derefter præsenterer vi master problemet og subproblemet ved søjlegenereringsalgoritmen. Efterfølgende beskriver vi branch-and-bound strukturen og forskellige strategier til accelerering af branch-and-price metoder. Vi beskriver generaliseringer af problemet og rapporterer beregningsmæssige resultater for de klassiske Solomon test problemer. Afslutningsvist præsenterer vi vore konklusioner og diskuterer åbne problemstillinger.

Introductory remarks

This thesis is not really meant to be read from cover to cover. Instead, the material is organized in four independent papers. This type of organization should not get in the way of a reader with some acquaintance with the subject, however, in this section it is described just what the contributions of each paper are and how the papers are related. Paper 1 and paper 4 are survey papers. In paper 1 an attempt has been made to provide a complete survey of the exact approaches for the VRPTW. The focus of paper 4 is on the path formulation of the VRPTW and column generation. Paper 2 and paper 3 represent the main part of the work during my graduate studies and describe the development of respectively a dual algorithm and a branch and cut algorithm for the solution of the VRPTW.

Paper 1: Formulations and exact approaches for the vehicle routing problem with time windows

Although this is the first paper of the thesis, it is actually the last paper I wrote during my graduate studies and it represents my view on the research of this field in the fall of 2005. The purpose of this paper is to provide a complete survey of the exact approaches proposed in the last three decades for the solution of the VRPTW. The material in the survey is strictly organized according to four different formulations of the VRPTW. This is somewhat different from other surveys and in my opinion one of the contributions of the survey. The relationship between the research on the VRPTW and the TSP is discussed because of the fundamental role of the TSP in combinatorial optimization in general and routing and scheduling in particular. I have included some historical references in this survey that I believe improve the understanding of this field. The survey includes new material related to the polyhedral approach that is not described in other surveys. In particular it presents my recent work with Boland and Madsen, which is the subject of paper 3 of this thesis. I also believe that the treatment of the spanning tree formulation of the VRPTW is more thorough than elsewhere and it illustrates that further work is required in this area. Finally, the survey presents the research on the decomposition approach of the path formulation including my work on acceleration strategies for the Lagrangian dual problem with Larsen and Madsen, which is the subject of paper 2 of this thesis. A question that the survey also discusses is this: What are the gaps between the bounds of the elementary path formulation and the non-elementary relaxation with 2-cycle elimination? I made some measurements in relation to this question that are included in the survey and I believe they are interesting because they made a clear case for eliminating higher order of cycles. Following this the 2-cycle elimination scheme was generalized by other authors. The survey concludes by discussing future directions of research that in my opinion are not as clearly expressed elsewhere.

Paper 2: Lagrangian duality applied to the vehicle routing problem with time windows

This is the first paper I wrote during my graduate studies and it is an extension of the work for my Masters' thesis. The topic of my Masters' thesis was also Lagrangian relaxation and its application to the VRPTW; particularly how the Lagrangian multipliers could be determined in an efficient way. I was motivated by the work on acceleration strategies at the master problem level by Kohl and Madsen. The authors had implemented a method for solving the Lagrangian dual problem that combined the use of the subgradient algorithm and a version of the bundle algorithm but had only applied the method to a set of test problems with clustered customer locations. During my Masters' thesis I had access to a software package for the solution of convex nondifferentiable optimization problems. I therefore only had to provide a subroutine for evaluating a single subgradient at each trial point. In the Lagrangian relaxation of the VRPTW I considered this corresponds to solving a resource-constrained shortest path problem. Larsen had just completed the development of a state-of-the-art column generation algorithm for the VRPTW and provided me with the subroutine for solving the shortest path problem. I applied this dual algorithm to all of the Solomon problem classes including the problems with wide time windows and

the results were promising. The method created easier shortest path subproblems because one could start with small values of the Lagrangian multipliers and increase them to the optimal level. This is different from standard column generation were one typically finds an initial feasible basis with high objective function coefficients which in turn gives high dual values. Another drawback of standard column generation is instability of the solution. Instability refers to a situation where the current solution is closer (with respect to some norm) to the optimal solution than the next solution. Identifying the difficulty of the subproblem relative to the behavior of the Lagrangian multipliers and addressing the drawbacks of standard column generation was the main contributions of the Masters' thesis. In order to find integer solutions we combined the dual algorithm with Larsen's column generation algorithm creating a hybrid algorithm. However, in 2000 it turned out that we could not use the package for nondifferentiable optimization and we therefore developed our own trust-region method for maximizing concave piece-wise linear functions, which are encountered in Lagrangian relaxation of integer linear programming problems. The development of this algorithm is the subject of paper 2. Kaj Madsen was also involved in the development of the trust-region method, which was based on some of his previous work in nonlinear optimization. We embedded the trust-region method in the branch and bound framework ABACUS. The dual algorithm is a row generation algorithm similar to polyhedral approaches, however, it is concerned with the characterization of the objective function of the combinatorial optimization problem instead of its polytope. Trust-region methods and polyhedral methods are cutting-plane algorithms and we denoted our method a Lagrangian branch and cut algorithm where the cutting-planes corresponds to the subgradients. We also introduced valid inequalities for the VRPTW polytope in the master problem but because the master problem is stated on the dual variables these inequalities are added as columns to the problem through a pricing step. One of the benefits of the trust-region method was that we avoided solving the quadratic problems of the bundle algorithms previously applied. It is also interesting to note that Thienel, the creator of ABACUS, thought that it would require a generalization of ABACUS to use the system for Lagrangian relaxation but we showed that by remaining within the context of linear programming when solving the Lagrangian dual problem it was already possible to embed Lagrangian relaxation in the system. The main contribution of paper 2 is the development of a trust-region method for solving the Lagrangian dual problem and embedding the trust-region method in a branch and bound algorithm. The method of paper 2 gives a significant speed-up compared to algorithms in the literature based on standard column generation and we also succeeded in solving two large-scale problems with 400 and 1000 customers, which to date are the largest problems ever solved to optimality.

Paper 3: Path inequalities for the vehicle routing problem with time windows

Through September 2001 to March 2002 I visited Dr. N. Boland of the University of Melbourne and paper 3 grew out of the work I carried out during that visit. The work was inspired by the work of Boland and Mak on polyhedral approaches for a variant of the traveling salesman problem. Because of the links to the TSP the material of paper 3 very much reflects the research on exact approaches for the TSP. We proposed a new formulation of the VRPTW that is based on a formulation of the TSP with time windows. We introduced new and stronger path inequalities for modeling the time windows of the VRPTW based on the polyhedral results of Mak for the TSP with replenishment constraints. We presented the first polyhedral results for the VRPTW by determining the dimension of the polytope and proving that the new path inequalities under certain conditions are facet defining. Furthermore, we take advantage of the precedence structure that the time windows induce and transfer precedence inequalities from the TSP context to the VRPTW. Finally, we make use of classes of inequalities for the ATSP in our implementation of a branch and cut algorithm. The contributions of paper 3 are therefore both theoretical and algorithmic, i.e. it is the first branch and cut algorithm for this variant of the VRPTW. The computational results of our algorithm shows that the polyhedral approach is a promising direction of research and in my opinion the conclusion is that the path formulation is no longer the unchallenged winning strategy for solving the VRPTW. However, it is clear that the amount of research effort spent to solve VRPTW by the polyhedral

approach is not comparable with what has been dedicated to the decomposition approach of the path formulation. Paper 3 is only the beginning of the development of the polyhedral approach for the VRPTW and a better understanding of its polytope and further work on developing efficient separation routines should yield much better computational results than those reported here.

Paper 4: Vehicle routing problem with time windows

In 2005 GERAD celebrated its 25th anniversary. The Gencol team is one of the best known research teams of the center and has made significant advances in the integer programming column generation area. The group originally focused on the vehicle routing problem with time windows and has made very substantial achievements in this area so it was a great privilege to contribute to a chapter on the VRPTW for the 25th GERAD anniversary volume on column generation. Paper 4 represents this contribution to the book on column generation edited by Desaulniers, Desrosiers, and Solomon, all part of the Gencol team. Paper 4 focuses on the methodological evolution, including cutting planes, parallelism, acceleration strategies for the master problem, novel subproblem approaches, and generalizations of the VRPTW. It also reports computational results for the classic Solomon test problems. It is clear that there is some overlap between paper 1 and paper 4 but in my opinion it is still valuable to include both surveys in this thesis because they are structured differently and with a different focus thereby highlighting different aspects of the research on the VRPTW.

Acknowledgements

I would like to express my deep thanks to my principal supervisor Professor Oli B. G. Madsen of the Centre for Traffic and Transport, Technical University of Denmark, whose support has been a mainstay since I started my graduate studies. I would also like to express my gratitude to my supervisor Associate Professor Jesper Larsen of the department of Informatics and Mathematical Modelling, Technical University of Denmark, whose advise and help over the course of my project has been extremely valuable. I would also like to thank my supervisor Professor Kaj Madsen of the department of Informatics and Mathematical Modelling, Technical University of Denmark, for all his suggestions regarding the development of a trust-region algorithm for the Lagrangian dual problem.

I am very grateful to Dr. Natashia Boland of University of Melbourne for giving me the opportunity to visit University of Melbourne through September 2001 to March 2002 and benefit from her broad research experience. I am also very grateful to Dr. Vicky Mak of Deakin University for all the fruitful discussions with her during my stay in Melbourne and subsequently. I feel very indebted to Boland and Mak for giving me the opportunity to benefit from their knowledge about polyhedral combinatorics.

The scholarship for my graduate studies was provided by the Technical University of Denmark. The research presented in this thesis was carried out in the Centre for Traffic and Transport of the Technical University of Denmark. I appreciate all the support these institutions offered.

January 10, 2006

Brian Kallehauge

Contents

Chapter 1

Formulations and exact approaches for the vehicle routing problem with time windows

Brian Kallehauge
Centre for Traffic and Transport, Technical University of Denmark

Abstract

In this paper we review the exact algorithms proposed in the last three decades for the solution of the vehicle routing problem with time windows (VRPTW). The exact approaches for the VRPTW are in many aspects inherited from work on the traveling salesman problem (TSP). In recognition of this fact this paper is structured relative to four seminal papers concerning the formulation and exact solution of the TSP, i.e. the arc formulation, the arc-node formulation, the spanning tree formulation, and the path formulation. We give a detailed analysis of the formulations of the VRPTW and a review of the literature related to the different formulations. There are two main lines of development in relation to the exact approaches for the VRPTW. One is concerned with the general decomposition approach and the solution to certain dual problems associated with the VRPTW. Another more recent direction is concerned with the analysis of the polyhedral structure of the VRPTW. We conclude by examining possible future lines of research in the area of the VRPTW.

1.1 Introduction

In 1959, a paper by Dantzig and Ramser [18] appeared in the journal Management Science concerning the routing of a fleet of gasoline delivery trucks between a bulk terminal and a number of service stations supplied by the terminal. The distance between any two locations is given and a demand for a certain product is specified for the service stations. The problem is to assign service stations to trucks such that all station demands are satisfied and total mileage covered by the fleet is minimized. The authors imposed the additional conditions that each service station is visited by exactly one truck and that the total demand of the stations supplied by a certain truck does not exceed the capacity of the truck. The problem formulated in the paper by Dantzig and Ramser [18] was given the name 'truck dispatching problem'. I do not know who coined the name 'vehicle routing problem' (VRP) for Dantzig and Ramser's

problem but it caught on in the literature and is the title of the most recent book on the problem, and some of its main variants, edited by Toth and Vigo [85]. In this book, Toth and Vigo [86] considered branch and bound algorithms for the VRP, Naddef and Rinaldi [72] branch and cut algorithms for the VRP and polyhedral studies, Bramel and Simchi-Levi [8] set covering based approaches for the VRP, Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis [15] the VRP with time windows, Toth and Vigo [87] the VRP with backhauls, and Desaulniers, Desrosiers, Erdmann, Solomon, and Soumis [20] the VRP with pickup and delivery. Furthermore, the book reviews heuristic approaches and issues arising in real-world applications. Now the basic version of the VRP is often given the name 'capacitated vehicle routing problem' (CVRP) to distinguish it from its variants. In this paper we consider the variant of the VRP with time windows (VRPTW), where each customer must be visited within a specified time interval, called a time window. We consider the case of hard time windows where a vehicle must wait if it arrives before the customer is ready for service and it is not allowed to arrive late. In the case of soft time windows a violation of the time window constraints is accepted but then a price must be paid.

Dantzig and Ramser [18] described how the VRP may be considered as a generalization of the traveling salesman problem (TSP). They described the generalization of the TSP with multiple salesmen and called this problem the 'clover leaf problem', a name that is the very picture of the problem. If there are m salesmen we will refer to the clover leaf problem as the m-TSP, a less lucid name. If in the m-TSP we impose the condition that specified deliveries be made at every location, excepting the start location, we get Dantzig and Ramser's problem. Obviously the VRP is identical with the m-TSP if the total demand of all locations is less than the capacity of a single vehicle. The standard reference book on the TSP was edited by Lawler, Lenstra, Rinnooy Kan, and Shmoys [65]. In this book Hoffman and Wolfe [42] describe how the importance of the TSP comes from the fact that it is typical of other problems of its genre: combinatorial optimization.

Dantzig had previously collaborated with Fulkerson and Johnson in developing an exact approach to the TSP. The appearance of their paper 'Solution of a large-scale traveling-salesman problem' (Dantzig, Fulkerson, and Johnson, 1954) in the journal Operations Research was according to Hoffman and Wolfe [42] "one of the principal events in the history of combinatorial optimization". In this paper the authors first associated with every tour a vector whose entries are indexed by the roads between the cities. An entry of this vector is 1 whenever the road between a pair of cities is traveled, otherwise it is 0. They also defined the linear equations that ensure all cities are visited exactly once in all representations of tours. These equations are called the degree constraints. Second, they defined a linear objective function that expressed the cost of a tour as the sum of road distances of successive pairs of cities in the tour. The problem is then to minimize the linear objective function such that the degree constraints are satisfied and the solution forms a tour. Third, the authors made a linear programming problem out of this integer programming problem by identifying just enough additional linear constraints on the vectors to assure that the minimum is assumed by some tour. This lead to the introduction of the subtour elimination constraints, which excludes solutions where cities are visited exactly once, but in a set of disconnected subtours. However, the authors pointed out that there are other types of constraints which sometimes must be added in addition to subtour elimination constraints in order to exclude solutions vectors involving fractional entries.

By now the approach of Dantzig, Fulkerson, and Johnson is basic in combinatorial optimization. The approach is concerned with identifying linear inequalities or cutting planes describing the polytope defined by the convex full of the points in the Euclidean space that represents the set of feasible solutions of the combinatorial optimization problem. No full description of the TSP polytope is known and because the TSP belongs to the class of NP-complete combinatorial optimization problems there is no hope for a polynomial-time cutting plane method for the TSP, unless NP = P. However, as Dantzig, Fulkerson, and Johnson showed the cutting plane approach can still be applied to the TSP by including the TSP polytope in a larger polytope (a relaxation) over which we minimize a linear objective function. In this way the TSP is formulated as a linear program that gives a lower bound for the TSP which can be useful in a

branch and bound algorithm. Padberg and Rinaldi [75] refined the integration of the enumeration approach of classical branch and bound algorithms with the polyhedral approach of cutting planes to create the solution technique called branch and cut. This method has been very successful in solving large-scale instances of the TSP and different authors have therefore applied the polyhedral approach to other hard combinatorial optimization problems. Laporte, Nobert, and Desrochers [62] were the first to apply the polyhedral approach to the VRP. Finally, we note that the field of discrete mathematics where combinatorial optimization problems are formulated as linear programs is called polyhedral combinatorics and we refer to the recent work of Schrijver [80] for a detailed treatment of this subject. For a treatment of polyhedral theory we refer to Nemhauser and Wolsey [73].

Now we consider another basic method in combinatorial optimization which is concerned with the characterization of the objective function of the combinatorial optimization problem instead of its polytope. Using relaxation and duality we can determine the optimal objective function value, or at least a good lower bound on it (assuming minimization), without explicitly solving the integer problem. In particular, we are concerned with Lagrangian relaxation and duality. A related technique is Dantzig-Wolfe decomposition, which provides an equivalent bound to the Lagrangian dual bound. In Lagrangian relaxation a set of complicating constraints are dualized into the objective function by associating Lagrangian multipliers with them. This gives us an infinite family of relaxations with respect to the Lagrangian multipliers. For a given set of values of the Lagrangian multipliers the relaxed problem is called the Lagrangian subproblem. The problem of determining the largest lower bound for this family is called the Lagrangian dual problem. A fundamental result in mathematical programming is that the Dantzig-Wolfe (generalized) linear programming problem of finding a convex combination of solutions to the (Lagrangian) subproblem that also satisfy the complicating constraints is dual to the Lagrangian dual problem. The book by Shapiro [81] marked the first appearance of the term Lagrangian relaxation in a textbook. In this book the treatment of duality takes a constructive rather than existential approach to Lagrangian multipliers. Everett [27] was the first to take this constructive point of view of Lagrangian multipliers, which is different from the Karush-Kuhn-Tucker point of view of optimality involving dual variables. For a treatment of Lagrangian duality we refer to Hiriart-Urruty and Lemaréchal [41, Chapter XII]. There exist two classical algorithms for solving the Lagrangian dual problem. The simplest algorithm for the Lagrangian dual problem is the subgradient algorithm. The other classical algorithm is the cutting-plane algorithm (a row-generation algorithm), which in the primal version is the column-generation algorithm. These algorithms are convex minimization algorithms and belong to the field of nonsmooth or nondifferentiable optimization. For a treatment of nonsmooth optimization we also refer to Hiriart-Urruty and Lemaréchal [41]. The combination of branch and bound and column generation was by analogy to branch and cut called branch and price by Savelsbergh [83]. Finally, when both variables and constraints are generated in the nodes of the search tree the procedure is called branch, cut, and price. In the last decade a number of frameworks for implementing branch, cut, and price has appeared, e.g. ABACUS [49], SYMPHONY [78], and BCP [13].

The use of Lagrangian relaxation in combinatorial optimization was in fact also inspired by the successful application of it to the TSP by Held and Karp [38, 39]. Lagrangian relaxation translates the problem of minimizing our objective function over a set of linear inequalities to finding the maximum of a concave piecewise affine function. There is a relationship between polyhedral combinatorics and Lagrangian relaxation. It is defined by the set of inequalities describing the convex hull of the incidence vectors of solutions to the (Lagrangian) subproblem. Held and Karp [38] proved using general linear programming theory that the maximization of the bound provided by the 1-tree Lagrangian relaxation of the TSP gives the same bound as the linear programming relaxation of the TSP proposed by Dantzig and Ramser [18] using the subtour inequalities. In this way the relationship between these two seminal contributions [18, 38, 39] is established, and thereby also an example of the relationship between polyhedral combinatorics and Lagrangian relaxation. If the complete set of inequalities of the subproblem was known it could be included in the integer programming problem and minimizing our objective function

over the set of complicating constraints and the inequalities of the subproblem would give the Lagrangian dual value.

The methods for the vehicle routing problem with time windows are in many aspects inherited from the work done for the traveling salesman problem. In recognition of this fact this paper is structured relative to the four seminal papers on the TSP formulations, i.e. the arc formulation, the arc-node formulation, the spanning tree formulation, and the path formulation. We only give a detailed analysis of the formulations in this paper but we do give a full review of the literature related to the different formulations. There are two main lines of development in relation to the exact approaches. One is concerned with the general decomposition approach and the solution to a certain dual problem associated with the primal VRPTW. Another direction is concerned with the analysis of the polyhedral structure of the VRPTW. The idea of convexity is central to both directions.

This paper is structured according to the four different formulations that have formed the starting point of the exact approaches to the VRPTW. The four formulations have also been considered in the context of the TSP. In what follows we give the complete list of references for the VRPTW relative to the four seminal papers on the TSP. We give the name of the authors of the papers concerning the TSP followed by a list of the papers on the VRPTW that consider a generalization of the approach for the TSP.

- Arc formulation, Dantzig, Fulkerson, and Johnson [17], [52, 69]
- Arc-node formulation, Miller, Tucker, and Zemlin [71], [4]
- Spanning tree formulation, Held and Karp [38, 39], [31]
- Path formulation, Houck, Picard, Queyranne, and Vemuganti [43], [60, 23, 37, 31, 58, 59, 63, 64, 14, 29, 9, 16, 28, 46, 54, 48]

Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis [15] and Kallehauge, Larsen, Madsen, and Solomon [53] also give recent surveys in relation to the VRPTW. The survey of Kallehauge et al. [53] is given in the context of column generation in general and the focus is therefore the path formulation of the VRPTW which has been studied by several authors. These surveys also give a status on the computational success of the state of the art algorithms proposed in the literature.

This paper is organized as follows. In Section 1.2 we define the VRPTW as a graph theoretic problem and introduce notation used throughout the paper. We also describe the complexity of the VRPTW and define its polytope. In Section 1.3 we consider the arc formulation of the VRPTW involving only binary variables associated with arcs of an underlying directed graph. In Section 1.4 we review the arc-node formulation of the VRPTW where we also associate variables with nodes of the directed graph. Section 1.5 considers a method to find lower bounds for the VRPTW, with the help of time and capacity constrained shortest spanning trees and Lagrangian relaxation. In Section 1.6 we consider a method to find lower bounds for the VRPTW, with the help of time and capacity constrained shortest paths and Lagrangian relaxation. Finally, in Section 1.7 we present some conclusions and discuss future directions of research.

1.2 Problem definition and notation

In this section we define the VRPTW as a graph theoretic problem and introduce notation used throughout the paper.

Definition 1.2.1 A time and capacity constrained digraph $D = (V, A, c, t, a, b, d, q)$ is defined by a node set $V = V_* \cup \{0, n+1\}$ for $V_* = \{1, \ldots, n\}$ the set of customer nodes and 0 and $n+1$ respectively the start and destination depot node, arc set $A = A_* \cup \delta^+(0) \cup \delta^-(n+1)$ for $A_* = A(V_*)$ the set of arcs spanned

by the customer nodes and $\delta^+(0) = \{(0,i) \mid i \in V_*\}$ the set of arcs leaving the start depot node and $\delta^-(n+1) = \{(i,n+1) \mid i \in V_*\}$ the set of arcs entering the destination depot node, costs on arcs $c \in \mathbb{Z}^A$ where $c_{ij} \leq c_{ik} + c_{kj}$ for $i,j,k \in V$, durations on arcs $t \in \mathbb{N}^A$ where $t_{ij} \leq t_{ik} + t_{kj}$ for $i,j,k \in V$, release and due times on nodes $a,b \in \{\mathbb{Z}_+ \cup \{+\infty\}\}^V$ where $a_0 = a_{n+1} = 0$, $b_0 = b_{n+1} = +\infty$, $a_i \geq t_{0i}$ and $b_i \geq a_i$ for $i \in V_*$ and $b_j \geq a_i + t_{ij}$ for $(i,j) \in A_*$, demands on nodes $d \in \mathbb{Z}_+^V$ where $d_0 = d_{n+1} = 0$, and capacity $q \in \mathbb{Z}_+$ where $q \geq d_i$ for $i \in V_*$ and $q \geq d_i + d_j$ for $(i,j) \in A_*$.

Definition 1.2.2 For any path $P = (v_1, \ldots, v_k)$ in D, the arrival times of the set of nodes $V(P)$ of the path is the vector $s \in \mathbb{Z}_+^{V(P)}$ defined by:

$$\begin{aligned} &s_{v_1} = a_{v_1}, \\ &s_{v_i} = \max\{s_{v_{i-1}} + t_{v_{i-1}v_i}, a_{v_i}\} \quad \text{for } i = 2, \ldots, k, \end{aligned}$$

the demand of the path is $d(V(P))$, and the cost of the path is $c(A(P))$.

Definition 1.2.3 We say that a path $P = (v_1, \ldots, v_k)$ in D is feasible if

$$s_{v_i} \leq b_{v_i} \quad \text{for } i \in V(P) \tag{1.1}$$

and

$$d(V(P)) \leq q. \tag{1.2}$$

Definition 1.2.4 We say that a path $P = (v_1, \ldots, v_k)$ in D is infeasible if

$$s_{v_i} > b_{v_i} \quad \text{for any } i \in V(P) \tag{1.3}$$

or

$$d(V(P)) > q. \tag{1.4}$$

Definition 1.2.5 An infeasible path $P = (v_1, \ldots, v_k)$ in D is said to be minimal infeasible if the subpaths of P induced by depriving $V(P)$ of respectively the starting node

$$V(P) \setminus \{v_1\} \tag{1.5}$$

and the end node

$$V(P) \setminus \{v_k\} \tag{1.6}$$

are feasible. We denote by $\mathscr{P}_D$ the set of all minimal infeasible paths in D.

Definition 1.2.6 A route in D is defined as a feasible path from 0 to $n+1$

$$R = (0, v_2, \ldots, v_{k-1}, n+1).$$

We denote by $\mathscr{R}$ the set of all routes in D.

Definition 1.2.7 A k-route in D is the union of k routes

$$\kappa_k = R_1 \cup R_2 \cup \cdots \cup R_k,$$

such that each node $v \in V_*$ belongs to exactly one set $V(R_i)$, $1 \leq i \leq k$. The cost of a k-route is $c(A(\kappa_k)) = c(\cup A(R_i) \mid i = 1, \ldots, k)$.

For any $W \subseteq V_*$, computing the number

$$k(W) = \min\{k \mid \text{a } \kappa_k \text{ exists in } D(W \cup \{0,1\})\}, \tag{1.7}$$

represents the problem of finding the minimum number of routes required to visit the subset of customer nodes W; we stress the fact that the notation $k(W)$ represents at the same time a number and a problem to solve.

Definition 1.2.8 A partition of the set of customer nodes V_* induced by a k-route κ_k is called a feasible k-partition. We denote by $K = \{k(V_*), \ldots, n\}$ the set of feasible partition sizes.

Definition 1.2.9 For each $k \in K$ we denote by $\mathscr{R}_k$ the set of k-routes with corresponding partition size k.

Definition 1.2.10 The set of all k-routes in D for $k \in K$ is denoted by $\mathscr{R}_K = \{\cup \mathscr{R}_k \mid k = k(V_*), \ldots, n\}$.

The vehicle routing problem with time windows is defined as follows. Given a time and capacity constrained digraph D, find a k-route of minimum cost, i.e.

$$\text{(VRPTW)} \quad \min\{c(A(\kappa_k)) \mid \kappa_k \in \mathscr{R}_K\}.$$

1.2.1 Complexity

If we place the restrictions on the instances of the VRPTW that $a_i = 0$ and $b_i = +\infty$ for every $i \in V_*$ the resulting restricted problem will be identical to the CVRP. If we furthermore place the restriction on the instances of the CVRP that $q \geq d(V_*)$ the resulting restricted problem will be identical to the TSP with multiple salesmen [71], sometimes denoted by m-TSP where m is the number of salesmen, and that can be transformed to the standard TSP [7]. On the other hand, the question of whether there exists a feasible solution for a given instance of the CVRP is an instance of the bin packing problem (BPP). Finally, if we place the restrictions on the instances of the VRPTW that $K = \{1\}$, $q \geq d(V_*)$, and $c_{ij} = 0$ for $(i,j) \in A$ the resulting restricted problem is the nonpreemptive single machine scheduling problem with release dates and deadlines (SS1). Garey and Johnson [32] proved the NP-completeness of TSP, BPP, and SS1. This implies by proof of restriction the NP-completeness of the VRPTW.

1.2.2 Polytope

Definition 1.2.11 With every k-route $\kappa_k \in \mathscr{R}_K$ in D, we associate an incidence vector $x^{\kappa_k} \in \mathbb{R}^A$ defined by:

$$x_{ij}^{\kappa_k} = \begin{cases} 1 & \text{if } (i,j) \in A(\kappa_k), \\ 0 & \text{if } (i,j) \notin A(\kappa_k). \end{cases}$$

Definition 1.2.12 The VRPTW polytope of a time and capacity constrained digraph $D = (V, A, c, t, a, b, d, q)$ is the convex hull of the incidence vectors of the k-routes in $\mathscr{R}_K$:

$$\mathscr{P}_{\text{VRPTW}} = \text{conv}\{x^{\kappa_k} \in \mathbb{R}^A \mid \kappa_k \in \mathscr{R}_K\}.$$

The vehicle routing problem with time windows is equivalent to minimizing the function $c^{\mathsf{T}}x$ over the VRPTW polytope. The NP-completeness of the VRPTW also implies that no description in terms of inequalities of the VRPTW polytope may be expected. However, polynomial-time computable lower bounds for the VRPTW can be obtained by including the VRPTW polytope in a larger polytope (a relaxation) over which $c^{\mathsf{T}}x$ can be minimized in polynomial time.

1.3 Subtour and path inequalities

In this section we consider a formulation of the VRPTW involving only binary variables associated with the arcs in D.

The VRPTW polytope is the set of those $x \in \mathbb{B}^A$ satisfying the degree equations

$$x(\delta^+(i)) = 1 \quad \text{for } i \in V_*, \tag{1.8}$$

$$x(\delta^-(i)) = 1 \quad \text{for } i \in V_*, \tag{1.9}$$

the subtour inequalities

$$x(A(W)) \leq |W| - 1 \quad \text{for } W \subseteq V_* \text{ with } |W| \geq 2, \tag{1.10}$$

and the path inequalities

$$x(A(P)) \leq |A(P)| - 1 \quad \text{for } P \in \mathscr{P}_D. \tag{1.11}$$

The formulation (1.8)-(1.11) of the VRPTW was proposed by Kallehauge, Boland, and Madsen [52]. The subtour inequalities were proposed by Dantzig, Fulkerson, and Johnson [17] in their seminal paper on the TSP. The idea of using path inequalities to model time window restrictions was presented by Ascheuer, Fischetti, and Grötschel [1] in their paper on the ATSPTW.

Laporte, Nobert, and Desrochers [62] generalized the subtour inequalities for the CVRP

$$x(A(W)) \leq |W| - \left\lceil \frac{d(W)}{q} \right\rceil \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset. \tag{1.12}$$

Naddef and Rinaldi [72] reviewed capacity inequalities of the CVRP polytope including the rounded capacity inequalities (1.12). Kohl, Desrosiers, Madsen, Solomon, and Soumis [59] further generalized the subtour inequalities for the VRPTW

$$x(A(W)) \leq |W| - k(W) \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset, \tag{1.13}$$

and denoted them k-path inequalities. If we in the formulation of the VRPTW replace the subtour inequalities (1.10) with the capacity inequalities (1.12) then it is sufficient to only include condition (1.1) and (1.3) in Definition 1.2.3 and Definition 1.2.4, respectively. Then we denote by $\mathscr{P}_D^{\mathrm{TW}}$ the set of minimal time infeasible paths of Definition 1.2.5 and redefine (1.11) to

$$x(A(P)) \leq |A(P)| - 1 \quad \text{for } P \in \mathscr{P}_D^{\mathrm{TW}}. \tag{1.14}$$

However, further replacing the capacity inequalities (1.12) with the k-path inequalities (1.13) is not sufficient to drop (1.14) in the formulation (1.8), (1.9), (1.12), and (1.14).

Kallehauge et al. [52] presented a class of strengthened path inequalities S_1 for the VRPTW based on the polyhedral results obtained by Mak [70] in the context of the asymmetric traveling salesman problem with replenishment arcs. Furthermore, Kallehauge et al. [52] determined the dimension of the VRPTW polytope and proved that the S_1 inequalities are facet defining under certain assumptions. These were the first polyhedral results for the VRPTW. Kallehauge et al. [52] also transferred the precedence constraints of Balas, Fischetti, and Pulleyblank [3] to the VRPTW context. Finally, the authors implemented a branch and cut algorithm that showed promising results and reported a solution to a previously unsolved 50-node test problem of Solomon [82].

Mak and Ernst [69] have also studied a formulation of the VRPTW similar to (1.8)-(1.11) and presented five new classes of valid inequalities for the VRPTW. The first four classes are based on the well-known D_k cycle inequalities [35] and the last is a class of path inequalities related to the S_1 class. The authors also proved that the new classes of inequalities are facet defining under certain assumptions.

1.4 Resource inequalities

Next we introduce a formulation of the VRPTW where we also associate variables with the nodes in D.

The integer solutions of (1.8)-(1.11) are exactly the incidence vectors of k-routes, so it gives an integer programming formulation of the VRPTW. The class of subtour inequalities (1.10) have a cardinality growing exponentially with n. An equivalent class of inequalities with polynomial cardinality was proposed by A. W. Tucker in 1960 [71]. He introduced node variables $u \in \mathbb{Z}^{V_*}$ and proposed the inequalities

$$u_i - u_j + px_{ij} \leq p-1 \quad \text{for } (i,j) \in A_*. \tag{1.15}$$

where $p \leq n$. The node variables u_i play the role of node potentials in an electrical network and the inequalities involving them serve to eliminate routes that do not begin at the start depot node 0 and end at the destination depot node $n+1$. This is already achieved by the subtour inequalities of Dantzig et al. [17]. The u_i can be adjusted so that $u_i = j$ if customer i is the jth customer visited in the route which includes customer i. The node variables u_i therefore represent the accumulated number of visits along a route. The inequalities (1.15) ensure that no more than p customers are visited in one route. For $p \geq n$ we have the standard VRPTW. In this way p is a resource limit on the number of visits in a route and we generally denote (1.15) resource inequalities.

Kulkarni and Bhave [61] generalized A. W. Tucker's inequalities for the CVRP, i.e. introduced a class equivalent to (1.12) but with polynomial cardinality. If every demand d_i for $i \in V_*$ represents a pick-up at customer i and $y \in \mathbb{Z}^{V_*}$ then

$$y_i - y_j + qx_{ij} \leq q - d_j \quad \text{for } (i,j) \in A_*, \tag{1.16}$$

where $d_i \leq y_i \leq q$ for $i \in V_*$, are denoted pick-up inequalities. For any route $R = (0, v_2, \ldots, v_{k-1}, n+1)$ where $k \geq 3$, the node variables of the route $y_{v_i} \in \mathbb{Z}$, $i = 2, \ldots, k-1$, can be adjusted so that:

$$y_{v_i} = \sum_{j=2}^{i} d_{v_j}, \tag{1.17}$$

where $y_{v_{k-1}} \leq q$. In case every demand d_i for $i \in V_*$ represents a delivery to customer i and $y' \in \mathbb{Z}^{V_*}$ then

$$y'_i - y'_j - qx_{ij} \geq d_j - q \quad \text{for } (i,j) \in A_*, \tag{1.18}$$

where $y_i \leq q - d_i$ for $i \in V_*$, are denoted delivery inequalities. For any route $R = (0, v_2, \ldots, v_{k-1}, n+1)$ where $k \geq 3$, the node variables of the route $y'_{v_i} \in \mathbb{Z}$, $i = 2, \ldots, k-1$, can be adjusted so that:

$$y'_{v_i} = d(V(R)) - \sum_{j=2}^{i} d_{v_j}, \tag{1.19}$$

where $y_{v_2} \leq q - d_{v_2}$. In the standard VRPTW it is required that all demands represent a pick-up or alternatively that all demands represent a delivery.

Desrochers and Laporte [22] further generalized A. W. Tucker's inequalities for the VRPTW, but in the case of time windows the resource inequalities are not only equivalent to the generalized subtour inequalities (1.13) of Dantzig et al. [17], but also equivalent to the path inequalities (1.14). If $s \in \mathbb{Z}^{V_*}$ then

$$s_i - s_j + (b_i + t_{ij} - a_j)x_{ij} \leq b_i - a_j \quad \text{for } (i,j) \in A_*, \tag{1.20}$$

where $a_i \leq s_i \leq b_i$ for $i \in V_*$, are denoted the time inequalities.
For any route $R = (0, v_2, \ldots, v_{k-1}, n+1)$ where $k \geq 3$, the node variables of the route $s_{v_i} \in \mathbb{Z}$, $i = 2, \ldots, k-1$, can be adjusted so that:

$$s_{v_1} = a_{v_1}, \tag{1.21}$$

$$s_{v_i} = \max\{s_{v_{i-1}} + t_{v_{i-1}v_i}, a_{v_i}\} \quad \text{for } i = 2, \ldots, k, \tag{1.22}$$

where $s_{v_i} \leq b_{v_i}$ for $i = 2, \ldots, k-1$.

The VRPTW polytope is the set of those $x \in \mathbb{B}^A$, $y \in \mathbb{Z}^{V_*}$, and $s \in \mathbb{Z}^{V_*}$ satisfying the degree equations

$$x(\delta^+(i)) = 1 \quad \text{for } i \in V_*, \tag{1.23}$$

$$x(\delta^-(i)) = 1 \quad \text{for } i \in V_*, \tag{1.24}$$

the pick-up inequalities

$$y_i - y_j + q x_{ij} \leq q - d_j \quad \text{for } (i,j) \in A_*, \tag{1.25}$$

the time inequalities

$$s_i - s_j + (b_i + t_{ij} - a_j) x_{ij} \leq b_i - a_j \quad \text{for } (i,j) \in A_*, \tag{1.26}$$

and the bounds

$$s_i \leq b_i \quad \text{for } i \in V_*, \tag{1.27}$$

$$s_i \geq a_i \quad \text{for } i \in V_*, \tag{1.28}$$

$$y_i \leq q \quad \text{for } i \in V_*, \tag{1.29}$$

$$y_i \geq d_i \quad \text{for } i \in V_*. \tag{1.30}$$

The formulation (1.23)-(1.30) of the VRPTW was proposed by Bard, Kontoravdis, and Yu [4]. However, the authors considered the problem (1.7) of finding the minimum number of routes required to visit the set of customers V_*. This problem is equivalent to minimizing the function $x(\delta^+(0))$ over the VRPTW polytope. Bard et al. [4] proposed the first branch and cut algorithm for the VRPTW based on this formulation. They considered a number of well-known inequalities from the TSP and VRP and proposed two new types of path inequalities taking into account the time windows of the problem. However, from a computational point of view the generalized subtour inequalities were the most effective. A reason for this is that the authors developed efficient heuristics for the separation of subtour inequalities. Furthermore, these path inequalities are quite weak compared to e.g. the S_1 inequalities proposed in the context of the VRPTW by Kallehauge et al. [52]. Bard et al. [4] also used a so-called greedy randomized adaptive search procedure (GRASP) for finding feasible solutions or upper bounds in the search tree. The branch and cut method of Bard et al. [4] showed promising computational results.

Formulating the integer programming model is only the first step when hard optimization problems are solved by branch and cut. The crucial part is the subset one considers of the finite family of defining inequalities of the associated polytope. It is well-known that A. W. Tucker's inequalities generally provide worse linear programming bounds than families of inequalities with exponential cardinality. In the context of the ATSPTW Ascheuer, Fischetti, and Grötschel [2] noted that their model involving only binary variables e.g. cannot handle as general objective functions as a model involving node variables. One example could be a makespan type of objective where the total time spent is minimized, i.e. including waiting time. Depending on the application this should therefore be the criterion for considering node variables or not because strengthened path inequalities dominate Tucker's inequalities.

1.5 Trees

In this section we consider a method to find lower bounds for the VRPTW, with the help of time and capacity constrained shortest spanning trees and Lagrangian relaxation.

Definition 1.5.1 A 0-arborescence in D is a subset B of A such that B forms a shortest spanning tree on $V \setminus \{n+1\}$ rooted at node 0 and such that for each node $v \in V_*$ there is a feasible path from 0 to v.

Definition 1.5.2 A routed arborescence, or just arborescence, in D is a subset T of A such that $T \setminus \delta^-(n+1)$ is a 0-arborescence and such that T contains a subset of arcs entering $n+1$, say $F = T \cap \delta^-(n+1)$, where $|F| = |T \cap \delta^+(0)|$. We denote by $\mathscr{T}$ the collection of all arborescences in D. For any arborescence T in D, the cost is defined by $c(T)$.

The shortest (= minimum cost) arborescence problem with time windows and capacity constraints is defined as follows. Given a time and capacity constrained digraph D, find an arborescence T of minimum cost, i.e.

(SAPTWCC) $\min\{c(T) \mid T \in \mathscr{T}\}$.

Papadimitriou [76] proved the NP-completeness of the capacitated tree problem and therefore by proof of restriction the SAPTWCC is also NP-complete.

Definition 1.5.3 With every arborescence $T \in \mathscr{T}$ in D, we associate an incidence vector $x^T \in \mathbb{R}^A$ defined by:

$$x_{ij}^T = \begin{cases} 1 & \text{if } (i,j) \in T, \\ 0 & \text{if } (i,j) \notin T. \end{cases}$$

Definition 1.5.4 The SAPTWCC polytope of a time and capacity constrained digraph D is the convex hull of the incidence vectors of the arborescences in $\mathscr{T}$:

$$\mathscr{P}_{\text{SAPTWCC}} = \text{conv}\{x^T \in \mathbb{R}^A \mid T \in \mathscr{T}\}.$$

The shortest arborescence problem with time windows and capacity constraints is equivalent to minimizing the function $c^{\mathsf{T}}x$ over the SAPTWCC polytope.

The SAPTWCC polytope is the set of those $x \in \mathbb{B}^A$ satisfying the indegree equations

$$x(\delta^-(i)) = 1 \quad \text{for } i \in V_*, \tag{1.31}$$

the cut-set inequalities

$$x(\delta^-(W)) \geq 1 \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset \tag{1.32}$$

the path inequalities

$$x(A(P)) \leq |A(P)| - 1 \quad \text{for } P \in \mathscr{P}_D \tag{1.33}$$

and the flow balance equation

$$x(\delta^+(0)) - x(\delta^-(n+1)) = 0. \tag{1.34}$$

The SAPTWCC can be solved by considering two separate problems:

- the determination of a shortest 0-arborescence B^* in D, defined by those $x \in \mathbb{B}^A$ satisfying (1.31), (1.32), and (1.33), and
- the determination of a subset F^* of minimum cost arcs entering the destination depot $n+1$, defined by those $x \in \mathbb{B}^A$ satisfying (1.34).

To see the relationship between the VRPTW and the SAPTWCC we consider a slightly different formulation of the VRPTW equivalent to (1.8)-(1.11). The outdegree (1.8) and indegree (1.9) equations give us a mean to alter the "inside" form of the subtour inequalities (1.10). By subtracting (1.8) from (1.10) we obtain the "outside" form defined by the set of arcs in D leaving W

$$x(\delta^+(W)) \geq 1 \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset, \tag{1.35}$$

and by subtracting (1.9) from (1.10) we obtain the outside form defined by the set of arcs in D entering W

$$x(\delta^-(W)) \geq 1 \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset. \tag{1.36}$$

Furthermore, the degree equations (1.8) and (1.9) together with the subtour inequalities (1.10) imply that

$$x(\delta^+(0)) - x(\delta^-(n+1)) = 0. \tag{1.37}$$

The formulation (1.8)-(1.11) of the VRPTW is therefore equivalent to

$$x(\delta^+(i)) = 1 \quad \text{for } i \in V_*, \tag{1.38}$$

$$x(\delta^-(i)) = 1 \quad \text{for } i \in V_*, \tag{1.39}$$

$$x(\delta^-(W)) \geq 1 \quad \text{for } W \subseteq V_* \text{ with } W \neq \emptyset, \tag{1.40}$$

$$x(A(P)) \leq |A(P)| - 1 \quad \text{for } P \in \mathscr{P}_D, \tag{1.41}$$

$$x(\delta^+(0)) - x(\delta^-(n+1)) = 0. \tag{1.42}$$

In the formulation (1.38)-(1.42) of the VRPTW, the outdegree equations (1.38) appear as the complicating constraints. If these constraints were not present the VRPTW would reduce to the SAPTWCC. To take advantage of this problem structure we therefore consider the Lagrangian relaxation with respect to the outdegree equations (1.38). For any $\lambda \in \mathbb{R}^{V_*}$ we consider the Lagrangian function defined by:

$$L(\lambda, x) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} - \sum_{i \in V_*} \lambda_i \left(\sum_{j \in V} x_{ij} - 1 \right) \tag{1.43}$$

$$= \tilde{c}^{\mathsf{T}} x + \lambda(V_*), \tag{1.44}$$

where

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \lambda_i & \text{for } i \in V_*, j \in V, \\ c_{ij} & \text{for } i = 0, j \in V_*. \end{cases} \tag{1.45}$$

The Lagrangian problem associated with λ is defined by:

$$z(\lambda) = \min \left\{ \tilde{c}^{\mathsf{T}} x \mid x \in \mathbb{B}^A \text{ satisfies conditions (1.39)} - \text{(1.42)} \right\} + \lambda(V_*) \tag{1.46}$$

where λ is fixed in $\mathbb{R}^{V_*}$. Problem (1.46) is a SAPTWCC with the cost function given by $\tilde{c} : A \to \mathbb{R}$.

The Lagrangian dual problem is defined by:

$$\max\{z(\lambda) \mid \lambda \in \mathbb{R}^{V_*}\}. \tag{1.47}$$

Definition 1.5.5 Let the set of arborescences $\mathscr{T}$ be indexed with $k = 1, \ldots, |\mathscr{T}|$ so T_k is the kth arborescence and define the cost of the kth arborescence

$$c_k = \sum_{(i,j) \in T_k} c_{ij} x_{ij}^{T_k}$$

and the outdegree of node $i \in V_*$ in the kth arborescence

$$a_{ik} = \sum_{j \in V} x_{ij}^{T_k} \quad \text{for } i = 1, \ldots, n.$$

The Lagrangian problem (1.46) is then defined by:

$$z(\lambda) = \min_{1 \leq k \leq |\mathscr{T}|} \left\{ c_k - a_k^{\mathsf{T}} \lambda \right\} + \lambda(V_*) \tag{1.48}$$

and the Lagrangian dual problem is defined by:

$$\max_{\lambda \in \mathbb{R}^{V_*}} \{ \min_{1 \leq k \leq |\mathscr{T}|} \left\{ c_k - a_k^{\mathsf{T}} \lambda \right\} + \lambda(V_*) \}. \tag{1.49}$$

Since $\mathscr{T}$ is finite it allows us to express (1.49) as the following linear program with many constraints or rows:

$$\begin{aligned} & \max \theta + \sum_{i \in V_*} \lambda_i \\ & \text{subject to} \\ & \theta \leqslant c_k - \sum_{i \in V_*} a_{ik} \lambda_i \quad \text{for } k = 1, \ldots, |\mathscr{T}|, \\ & \lambda_i \in \mathbb{R} \quad \text{for } i \in V_*, \\ & \theta \in \mathbb{R}. \end{aligned} \tag{1.50}$$

The LP dual of (1.50) is a linear program with many variables or columns:

$$\begin{aligned} & \min \sum_{k=1}^{|\mathscr{T}|} c_k y_k \\ & \text{subject to} \\ & \sum_{k=1}^{|\mathscr{T}|} a_{ik} y_k = 1 \quad \text{for } i \in V_*, \\ & \sum_{k=1}^{|\mathscr{T}|} y_k = 1, \\ & y_k \geqslant 0 \quad \text{for } k = 1, \ldots, |\mathscr{T}|. \end{aligned} \tag{1.51}$$

Problem (1.51) with y_k required to be integral is equivalent to the VRPTW. In case of integrality constraints an optimal solution to (1.51) must satisfy $y_{k^*} = 1$ for some $T_{k^*} \in \mathscr{T}$ and $y_k = 0$ for all $T_k \in \mathscr{T} \setminus \{T_{k^*}\}$. Problem (1.51) is the LP relaxation of the Dantzig-Wolfe decomposition obtained when any solution to the VRPTW is expressed as a non-negative convex combination of resource-constrained directed spanning trees. The relaxation of the VRPTW presented in this section has never been used directly in a branch and bound algorithm. The idea of using shortest spanning trees has been considered in the VRPTW context but only one paper [31] in the literature on the VRPTW has considered this classical approach in vehicle routing.

Held and Karp [38] explored the relationship between the symmetric and asymmetric traveling salesman problem and shortest spanning trees in undirected and directed graphs, respectively. Consider the symmetric TSP and the complete undirected graph $G = (V, E)$ on n nodes. A 1-tree is a subgraph T of G with nodes $1, 2, \ldots, n$ consisting of a tree on the nodes $2, 3, \ldots, n$ together with two edges incident with

node 1. In fact, a 1-tree T in G consists of exactly n arcs, whereas a tree in G consists of $n-1$ arcs. So a 1-tree is a tree with an additional arc added explaining the term 1-tree. A solution to the STSP is precisely a 1-tree in which all nodes have degree 2. The Lagrangian relaxation of the STSP with respect to the degree constraints $x(\delta(v)) = 2$, $v \in V \setminus \{1\}$, gives a shortest 1-tree Lagrangian problem and the Lagrangian dual problem can be expressed as a pair of linear programs similar to (1.50) and (1.51). Held and Karp [38] gave a column generation method and an ascent method for finding the Lagrangian dual value. The column generation method was able to solve the program (1.51) for most problems with $n = 12$ and some problems with $13 \leq n \leq 20$. On larger problems the convergence was always too slow and the authors noted that this was consistent with the behavior of other column generation techniques at that time referring to the work of Gilmore and Gomory [34]. Held and Karp [38] also described how the approach for the symmetric TSP carry over to the asymmetric case. In this case the authors introduced a type of directed subgraph they called a 1-arborescence, defined as an arborescence (directed tree) rooted at node 1 plus an arc $(v,1)$ joining some node $v \in V \setminus \{1\}$ to node 1. We remark that the authors notion of a 1-arborescence is slightly different from our r-arborescence, which is defined as an arborescence rooted at node r. Held and Karp [38] also described the extension of the 1-tree approach to the m-STSP in which the degree at node 1 is $2m$. This would later be further generalized to the symmetric CVRP. Held and Karp [38] already added in the proof of their paper that a new method for computing the Lagrangian dual value would be presented in a sequel to the paper. The following paper [39] was a milestone in the subject of Lagrangian relaxation in integer programming. Held and Karp [39] successfully introduced what became known as the subgradient algorithm (a term introduced by Held, Wolfe, and Crowder [40]) and influenced future research dramatically [5]. In 1974 Geoffrion [33] coined the term Lagrangian relaxation to describe the method of Held and Karp [38, 39]. Because of the initial use of the subgradient algorithm, Lagrangian relaxation to some extent became synonymous with the subgradient algorithm, which is unfortunate because this algorithm is the simplest algorithm for concave maximization and suffers from several drawbacks [41]. Indeed the method is mainly attractive because it is so simple to implement.

The earliest generalization of the approach by Held and Karp [38] was proposed by Christofides, Mingozzi, and Toth [11] for the SCVRP based on the k-degree center tree (k-DCT) relaxation of the SCVRP. The approach allow for the possibility of single customer routes. Fisher [30] presented a different relaxation of the SCVRP using shortest k-trees. Consider the symmetric CVRP and the complete undirected graph $G = (V,E)$ on n nodes. A k-tree is a subgraph of G consisting of $n-1+k$ edges that span the n nodes. The degree of the depot node 1 is $2k$. As the name suggests this is a generalization of the 1-tree approach of Held and Karp [38]. The author dualized the capacity constraints (1.12) of Laporte, Nobert, and Desrochers [62] and solved the Lagrangian dual problem using the subgradient algorithm and generation of violated capacity constraints. The Lagrangian dual problem expressed as an LP similar to (1.50) is exponential in size since it has exponentially many variables as well as constraints corresponding to the number of capacity constraints and k-trees, respectively. Instead of using the subgradient algorithm a cut and column generation algorithm similar to the one proposed by Kallehauge, Larsen, and Madsen [54] could be used for solving the dual problem.

Fisher [30] also described an extension of his method to the VRPTW. He introduced path inequalities (1.11) in a formulation of the VRPTW and relaxed these to obtain the same k-tree Lagrangian problem considered for the CVRP. He did not report any computational results. The extension to time windows was developed with K. Jörnsten and O.B.G. Madsen and together Fisher, Jörnsten, and Madsen [31] later presented computational results using the k-tree method. However, the shortest tree relaxation of the VRPTW has not been the subject of the same amount of research as the shortest path relaxation and in our view it is at this point an open question whether the formulation described in this section is effective.

We do not consider a formulation of the SAPTWCC with a fixed degree at the root node because we consider the variant of the VRPTW with a free number of vehicles. This is different from the TSP (VRP) where the authors consider rooted trees with a degree constraint at the root node because they consider problems with a fixed number of tours (routes). Toth and Vigo [84] proposed an algorithm for the shortest

arborescence problem with capacity constraints (SAPCC), however the extension to time windows has not been considered in the literature. Finally, we refer to Schrijver [80, Section 50.6a] for a complexity survey for shortest spanning trees.

1.6 Paths

We next consider a method to find lower bounds for the VRPTW, with the help of time and capacity constrained shortest paths and Lagrangian relaxation.

Definition 1.6.1 We denote by $D_{0,n+1}$ the time and capacity constrained digraph obtained from D by adding the arc $(0,n+1)$ to the set of arcs A of D. For notational convenience we denote the extended set of arcs by A. The cost $c_{0,n+1}$ and duration $t_{0,n+1}$ on arc $(0,n+1)$ is zero.

The elementary shortest path problem with time windows and capacity constraints is defined as follows. Given a time and capacity constrained digraph $D_{0,n+1}$, find a path from 0 to $n+1$ of minimum cost, i.e.

(ESPPTWCC) $\min\{c(A(R)) \mid R \in \mathscr{R}\}$.

Dror [26] proved the NP-completeness of the ESPPTWCC. Next we define a relaxation of the ESPPTWCC. A walk in $D_{0,n+1}$ from 0 to $n+1$ is a sequence of m nodes

$$R_{(k)} = (0, v_2, \ldots, v_{m-1}, n+1), \tag{1.52}$$

where $v_2, \ldots, v_{m-1}$ are not necessarily distinct. If $v_{i+p} \neq v_i$ for $2 \leq p \leq k$ the walk is called a non-elementary route with no k-cycles. The (non-elementary) shortest path problem with time windows and capacity constraints and no k-cycles is defined as follows. Given a time and capacity constrained digraph $D_{0,n+1}$, find a walk from 0 to $n+1$ of minimum cost containing no k-cycles, i.e.

(k-SPPTWCC) $\min\{c(A(R_{(k)})) \mid R_{(k)} \in \mathscr{R}_{(k)}\}$.

Pseudo-polynomial time algorithms exist for the k-SPPTWCC [21, 43, 23, 46]. If no cycle elimination is performed the problem is called the SPPTWCC. In $D_{0,n+1}$ we have that $\mathscr{R}_{(2)} \supseteq \mathscr{R}_{(3)} \supseteq \cdots \supseteq \mathscr{R}_{(n-1)} \supseteq \mathscr{R}$.

Definition 1.6.2 With every route $R \in \mathscr{R}$ in $D_{0,n+1}$, we associate an incidence vector $x^R \in \mathbb{R}^A$ defined by:

$$x_{ij}^R = \begin{cases} 1 & \text{if } (i,j) \in A(R), \\ 0 & \text{if } (i,j) \notin A(R). \end{cases}$$

Definition 1.6.3 The ESPPTWCC polytope of a time and capacity constrained digraph $D_{0,n+1}$ is the convex hull of the incidence vectors of the routes in $\mathscr{R}$:

$$\mathscr{P}_{\text{ESPPTWCC}} = \text{conv}\{x^R \in \mathbb{R}^A \mid R \in \mathscr{R}\}.$$

The elementary shortest path problem with time windows and capacity constraints is equivalent to minimizing the function $c^{\mathsf{T}}x$ over the ESPPTWCC polytope.

The ESPPTWCC polytope is the set of those $x \in \mathbb{B}^A$ satisfying the degree equations

$$x(\delta^+(0)) = 1, \tag{1.53}$$

$$x(\delta^-(n+1)) = 1, \tag{1.54}$$

the balance equations

$$x(\delta^-(i)) - x(\delta^+(i)) = 0, \quad \text{for } i \in V_*, \tag{1.55}$$

the subtour inequalities

$$x(A(W)) \leq |W| - 1 \quad \text{for } W \subseteq V_* \text{ with } |W| \geq 2, \tag{1.56}$$

and the path inequalities

$$x(A(P)) \leq |A(P)| - 1 \quad \text{for } P \in \mathscr{P}_D. \tag{1.57}$$

To see the relationship between the VRPTW and the ESPPTWCC we define the VRPTW in a slightly different way. Given a time and capacity constrained digraph $D_{0,n+1}$, find a collection of routes $\{R_k \mid k = 1,\ldots,n\}$ of minimum cost such that each node $v \in V_*$ is visited exactly once, i.e.

$$\begin{aligned} &\min \sum_{1\leq k\leq n} c(A(R_k)) \\ &\text{subject to} \\ &|\cup_{1\leq k\leq n} A(R_k) \cap \delta^+(v)| = 1 \quad \text{for } v \in V_*, \\ &R_k \in \mathscr{R} \quad \text{for } k = 1,\ldots,n. \end{aligned} \tag{1.58}$$

Remark 1.6.1 Since we have introduced the arc $(0, n+1)$ in $D_{0,n+1}$ the solution to (1.58) may contain a number of zero-cost routes not visiting any customer nodes and problem (1.58) is therefore equivalent to the VRPTW.

Definition 1.6.4 With every collection of routes $\kappa_n = \{R_k \mid k = 1,\ldots,n\}$ in $D_{0,n+1}$, we associate an incidence vector $x^{\kappa_n} \in \mathbb{R}^{A^n}$ defined by:

$$x_{ijk}^{\kappa_n} = \begin{cases} 1 & \text{if } (i,j) \in A(R_k), \\ 0 & \text{if } (i,j) \notin A(R_k). \end{cases}$$

The polytope of (1.58) is the set of those $x \in \mathbb{B}^{A^n}$ where $x_k \in \mathbb{B}^A$ satisfy (1.53)-(1.57) for $k = 1,\ldots,n$ and:

$$\sum_{1\leq k\leq n} \sum_{j\in V} x_{ijk} = 1 \quad \text{for } i \in V_*. \tag{1.59}$$

The vehicle routing problem with time windows is therefore equivalent to minimizing the function $\sum_{1\leq k\leq n} c^{\mathsf{T}} x_k$ over the polytope of (1.58).

In the formulation (1.53)-(1.59) of the VRPTW, the constraints (1.59) appear as coupling constraints, which link the individual variables x_k. If these constraints were not present the VRPTW would reduce to n ESPPTWCC problems, each with the simpler formulation (1.53)-(1.57), and thus become considerable simpler. To take advantage of this problem structure we therefore consider the Lagrangian relaxation with respect to the constraints (1.59). For any $\lambda \in \mathbb{R}^{V_*}$ we consider the Lagrangian function defined by:

$$L(\lambda, x) = \sum_{1\leq k\leq n} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ijk} - \sum_{i\in V_*} \lambda_i \left(\sum_{1\leq k\leq n} \sum_{j\in V} x_{ij} - 1 \right) \tag{1.60}$$

$$= \sum_{1\leq k\leq n} \tilde{c}^{\mathsf{T}} x_k + \lambda(V_*), \tag{1.61}$$

where

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \lambda_i & \text{for } i \in V_*, j \in V, \\ c_{ij} & \text{for } i = 0, j \in V_*. \end{cases} \tag{1.62}$$

The Lagrangian problem associated with λ is defined by:

$$z(\lambda) = \min\{ \sum_{1 \le k \le n} \tilde{c}^{\mathsf{T}} x_k \mid x_k \in \mathbb{B}^A \text{ satisfies conditions (1.53)} - \text{(1.57) for } k = 1, \ldots, n\} + \lambda(V_*) \tag{1.63}$$

where λ is fixed in $\mathbb{R}^{V_*}$. Problem (1.63) can be solved by considering n ESPPTWCC problems with the cost function $\tilde{c} : A \to \mathbb{R}$. Since the n ESPPTWCC subproblems are identical, one only needs to consider one subproblem and the Lagrangian problem takes the form:

$$z(\lambda) = n \min\{\tilde{c}^{\mathsf{T}} x \mid x \in \mathbb{B}^A \text{ satisfies conditions (1.53)} - \text{(1.57)}\} + \lambda(V_*) \tag{1.64}$$

The Lagrangian dual problem is defined by:

$$\max\{z(\lambda) \mid \lambda \in \mathbb{R}^{V_*}\}. \tag{1.65}$$

Definition 1.6.5 Let the set of routes $\mathscr{R}$ be indexed with $k = 1, \ldots, |\mathscr{R}|$ so R_k is the kth route and define the cost of the kth route

$$c_k = \sum_{(i,j) \in A(R_k)} c_{ij} x_{ij}^{R_k}$$

and the number of times node $i \in V_*$ is visited by the kth route

$$a_{ik} = \sum_{j \in V} x_{ij}^{R_k} \quad \text{for } i = 1, \ldots, n.$$

The Lagrangian problem (1.64) is then defined by:

$$z(\lambda) = n \min_{1 \le k \le |\mathscr{R}|} \{c_k - a_k^{\mathsf{T}} \lambda\} + \lambda(V_*) \tag{1.66}$$

and the Lagrangian dual problem is defined by:

$$z_{\mathrm{LD}}(\mathscr{R}) = \max_{\lambda \in \mathbb{R}^{V_*}} \{n \min_{1 \le k \le |\mathscr{R}|} \{c_k - a_k^{\mathsf{T}} \lambda\} + \lambda(V_*)\}. \tag{1.67}$$

In (1.67) we have that $z_{\mathrm{LD}}(\mathscr{R}_{(2)}) \le z_{\mathrm{LD}}(\mathscr{R}_{(3)}) \le \cdots \le z_{\mathrm{LD}}(\mathscr{R}_{(n-1)}) \le z_{\mathrm{LD}}(\mathscr{R})$.

Since $\mathscr{R}$ is finite it allows us to express (1.67) as the following linear program with many constraints or rows:

$$\begin{aligned} & \max n\theta + \sum_{i \in V_*} \lambda_i \\ & \text{subject to} \\ & \theta \leqslant c_k - \sum_{i \in V_*} a_{ik} \lambda_i \quad \text{for } k = 1, \ldots, |\mathscr{R}|, \\ & \lambda_i \in \mathbb{R} \quad \text{for } i \in V_*, \\ & \theta \in \mathbb{R}. \end{aligned} \tag{1.68}$$

The LP dual of (1.68) is a linear program with many variables or columns:

$$
\begin{aligned}
(1.69) \qquad & \min \sum_{k=1}^{|\mathscr{R}|} c_k y_k \\
& \text{subject to} \\
& \sum_{k=1}^{|\mathscr{R}|} a_{ik} y_k = 1 \quad \text{for } i \in V_*, \\
& \sum_{k=1}^{|\mathscr{R}|} y_k = n, \\
& y_k \geqslant 0 \quad \text{for } k = 1, \ldots, |\mathscr{R}|.
\end{aligned}
$$

Problem (1.69) with y_k required to be integral is equivalent to the VRPTW. This also holds true if we formulate the linear program with respect to $\mathscr{R}_{(k)}$ for some $2 \leq k \leq n-1$. Problem (1.69) is the LP relaxation of the Dantzig-Wolfe decomposition obtained when any solution to the VRPTW is expressed as a non-negative convex combination of routes, however, because the subproblems are identical the convexity constraints have been aggregated [53]. The aggregated formulation is equivalent to the standard set-partitioning formulation of vehicle routing problems [8]. There is a benefit in working with the decomposed formulation; it does not suffer from the drawback of symmetry present in the original formulation using the three-index variables x_{ijk} where a given solution can be represented in several ways by permuting the k indexing. The formulation of the VRPTW presented in this section using three-index variables have never been used in a branch and bound algorithm. Problem (1.69) have been an important construct in the formulation of algorithms for the VRPTW but more recently the dual point of view of (1.68) has also been considered.

Houck, Picard, Queyranne, and Vemuganti [43] presented a relaxation of the symmetric and asymmetric traveling salesman problem based on resource-constrained paths. The resource in their path definition is the number of arcs contained in the path and the limit on the consumption of this resource is the number of nodes in the graph n. They called a path containing n arcs an n-path. They called a path elementary if its nodes are all distinct except possibly the first and last node. If we fix a node 1 of the (directed) graph an n-path from node 1 to node 1 is a Hamilton tour if and only if it is elementary. The problem of finding an elementary n-path of minimum cost from node 1 to node 1 is equivalent to the traveling salesman problem and hence NP-complete. The authors therefore relaxed the condition that the path should be elementary. If $P = (1, i_1, \ldots, i_{n-1}, 1)$ denotes an n-path and there exists a k such that $i_k = i_{k+2}$ for some k then the path P is said to contain a 2-cycle. It was observed that many n-paths contained 2-cycles. They proposed a tighter relaxation by forbidding paths containing 2-cycles and presented a dynamic programming algorithm for finding an n-path of minimum cost which does not contain 2-cycles. Houck et al. [43] also showed that the problem of maximizing the bound derived by this relaxation could be expressed as linear programs similar to (1.68) and (1.69). They proposed a column generation scheme for solving the master problem but noted that the column generation was very slow in converging to the optimal solution. This is a typical observation in early research involving column generation. An American pioneer in linear programming computing techniques, Orchard-Hays [74, 34, p. 240] said: "Nevertheless, the D-W (Dantzig-Wolfe) (generalized programming) algorithm presents difficulties, and overall experience with its use has been mixed. This has led to some disappointment with decomposition algorithms". Houck et al. [43] then proposed to use the subgradient algorithm following the earlier approach of Held and Karp [39] and embedded this method in a branch and bound algorithm. The authors made an important concluding remark that the n-path relaxation can easily accommodate extra conditions. In fact the computational work required in the dynamic programming algorithm is just reduced when additional constraints are handled. In relation to this observation Picard and Queyranne [77] had previously considered

a time-constrained variant of the TSP.

Christofides, Mingozzi, and Toth [11, 12] generalized the n-path relaxation of Houck et al. [43]. They formulated the capacitated vehicle routing problem using resource-constrained paths where the resource is the accumulated demand q along the path and called these paths q-paths. They also considered the set-partitioning formulation of the CVRP similar to (1.69) but instead of solving the linear programming relaxation of this master problem they proposed a relaxation which could be solved by dynamic programming. In this way their method is a two-level dynamic programming approach. In 1987 Kolen, Rinnooy Kan, and Trienekens [60] introduced the first method for the exact solution of the VRPTW. Kolen et al. [60] extended the two-level dynamic programming approach of Christofides et al. for the VRPTW by introducing the accumulated time along the paths as an additional resource. The importance of this research comes from the introduction of the shortest path problem with time windows and capacity constraints SPPTWCC, which has played a prominent role in the research on the VRPTW. The method of Kolen et al. [60] was only capable of solving problems with up to 15 customers. The reason for this is the relaxation of the master level problem and the use of dynamic programming for solving the master problem.

The appearance of 'A new optimization algorithm for the vehicle routing problem with time windows' (Desrochers, Desrosiers, and Solomon, 1992) in the journal Operations Research was a breakthrough in the history of the VRPTW and furthermore an important paper in relation to the successful application of Dantzig-Wolfe decomposition and column generation in general. The method of Desrochers, Desrosiers, and Solomon is also based on the resource-constrained path formulation of the VRPTW but they used column generation to solve the linear programming relaxation of the set-partitioning master problem (1.69). The idea of embedding column generation in a branch and bound algorithm was previously introduced by Desrosiers, Soumis, and Desrochers [24] for the m-TSP with time windows. Another important contribution of Desrochers, Desrosiers, and Solomon was the introduction of the set of test problems developed by Solomon [82]. The introduction of a standard set of test problems is important because it enables relative evaluation of competing approaches. Of course different authors also need to consider the same problem variant and adhere to certain conventions with respect to the precision of problem data. Finally, the Desrochers, Desrosiers, and Solomon paper introduced the label setting algorithm of Desrochers [21] in the context of the VRPTW for solving the shortest path problem with time windows and capacity constraints. The authors extended the algorithm to include the 2-cycle elimination scheme of Houck, Picard, Queyranne, and Vemuganti [43]. Desrochers' algorithm has been an important component in the solution of a large class of resource constrained routing and scheduling problems [25] but as far as we are aware the manuscript 'An algorithm for the shortest path problem with resource constraints' (Desrochers, 1988) has not yet been published in an international journal.

Lagrangian decomposition is an approach that attempts to strengthen the bounds of Lagrangian relaxation [36]. This approach splits the original problem into two or more different types of subproblems. Halse [37] describes three different Lagrangian decompositions of the VRPTW: VS1, VS2, and VS3. In the VS1 decomposition the VRPTW is formulated using a shortest path problem with time windows (SPPTW) and a generalized assignment problem (GAP). The VS2 decomposition considers a shortest path problem with time windows and capacity constraints (SPPTWCC) and a semi-assignment problem (SAP). Finally, the VS3 decomposition considers an SPPTWCC and a GAP by including vehicle capacity constraints in both subproblems. The Lagrangian dual problems of the three Lagrangian decomposition approaches involves more multipliers than in the Lagrangian relaxation of the VRPTW (1.61). In fact, if n is the number of vehicles then the Lagrangian decompositions requires n times as many multipliers as the Lagrangian relaxation (1.61). Kohl [57] made an analytical comparison of the bounds provided by the Lagrangian decompositions and the Lagrangian relaxation and proved that the VS1 and VS2 decompositions give the same bound as the Lagrangian relaxation. Furthermore, he also proved that the VS3 decomposition gives the same bound as the Lagrangian relaxation under the assumptions that the vehicles are identical and a feasible solution exists for the VRPTW. This proof is non-trivial since the

two subproblems SPPTWCC and GAP do not have the integrality property. Since the Lagrangian decompositions increase the dimension of the Lagrangian dual problem, but do not strengthen the bound of the Lagrangian relaxation, the conclusion of the analysis by Kohl [57] is that the Lagrangian decompositions are not attractive for the identical-vehicle VRPTW. Fisher, Jörnsten, and Madsen [31] presented computational results based on the VS2 decomposition solving the Lagrangian dual problem using the subgradient algorithm. Although the Lagrangian decomposition considered by Fisher et al. [31] is not attractive compared to the Lagrangian relaxation the authors were the first to consider a path formulation of the VRPTW from the dual point of view. It would later become clear that it was the choice of the subgradient algorithm that impeded the dual approach.

Kohl and Madsen [58] proposed a method for the VRPTW based on the Lagrangian relaxation (1.61). They implemented a bundle algorithm of Lemaréchal, Strodiot, and Bihain [67] with an Euclidean steepest descent direction finding problem and the line-search of Lemaréchal [66] for determining the step-size. In relation to line-searches Hiriart-Urruty and Lemaréchal [41, p. 403] mentioned that "the modern tendency goes towards the so-called trust-region technique". A number of important issues were addressed by Kohl and Madsen [58]. First, the dimension of the Lagrangian dual problem is smaller in the Lagrangian relaxation than in the Lagrangian decompositions. Second, the convergence of the bundle algorithm is better compared to the subgradient algorithm. Third, using a bundle algorithm it is possible to obtain a primal solution equivalent to the variables of the Dantzig-Wolfe master problem (1.69). Finally, in the bundle algorithm one can choose a starting point with relatively small multiplier values and gradually increase the multipliers to the optimal level. The dual approach therefore creates easier shortest path subproblems because the modified arc costs are less negative, hence less negative cycles are introduced. Kohl and Madsen [58] only considered problems that required very little branching so a full computational study of the path formulation from a dual point of view was not performed.

Kohl [57] and Kohl, Desrosiers, Madsen, Solomon, and Soumis [59] addressed the need for improving the bounds provided by the non-elementary path formulation with 2-cycle elimination. They introduced the k-path inequalities (1.13) for $k = 2$ in the Dantzig-Wolfe master problem (1.69). The path formulation then has exponentially many variables as well as constraints and a column and cut generation approach was used for solving the master problem. They embedded the column and cut generation in a branch and bound algorithm and their work is one of the early examples of what became known as branch, price, and cut algorithms.

Larsen [63] parallelized the branch and bound search in the algorithm of Kohl et al. [59]. Furthermore, Larsen [63, 64] proposed column deletion and forced early stop for improving solution times. The column deletion procedure deletes columns in the master problem and is similar to the concept of the bundle reduction technique used by Kohl and Madsen [58], i.e. to limit the size of the coordinating master problem. The forced early stop terminates the SPPTWCC pricing algorithm as soon as one path with negative reduced cost is generated. The forced early stop is motivated from a dual point of view. In the column generation algorithm the initial master problem must be initialized with a feasible solution. If this primal solution corresponds to a dual solution with relatively high multiplier values compared to the optimal level then the difficulty of the SPPTWCC subproblem is relatively higher in the initial phase of the column generation algorithm. In the first iterations forced early stop therefore gives rapid improvements of the dual solution cutting down on solution times in the subproblem. Forced early stop is also called partial pricing whereas solving the subproblem to optimality is called full pricing.

Rich [79] and Cook and Rich [14] extended the work of Kohl et al. [59]. They proposed a new separation algorithm for the k-path inequalities for $k \geq 2$. The authors used the randomized algorithm given by Karger and Stein [55] that finds α-minimal cuts in undirected graphs in $O(n^{2\alpha}\log^2 n)$ time where n is the number of nodes. By setting $\alpha = k$, sets W for which $x(W) < k$ are found. If $k(W) \geq k$ in (1.7) then W induces a valid k-path inequality (1.13). Moreover, they parallelized the separation of k-path inequalities and the branch and bound search.

The bounds provided by the path formulation is also improved if we require the solution to the resource-constrained shortest path problem to be elementary. Beasley and Christofides [6] proposed to add a visitation resource for each node $v \in V_*$ with a lower and upper limit of zero and one, respectively. The visitation resource usage when passing through node v is one. This new resource definition ensures we generate only elementary paths. However, the size of the state space increases dramatically and Beasley and Christofides [6] expected that this formulation would only be suitable for solving problems with a small number of resources and made no further studies. The approach of Desrochers et al. [23] is attractive from a computational point of view because the SPPTWCC can be solved in pseudo-polynomial time. However, the method has the disadvantage of weakening the lower bound provided by the path formulation. In an effort to address this issue Feillet, Dejax, Gendreau, and Gueguen [29], Chabrier [9], and Danna and Le Pape [16] adapted the idea of Beasley and Christofides [6] to Desrochers' algorithm. The authors also generalized the consumption of the visitation resource. A node v is called unreachable with respect to a path P if the path already has visited v or there is no way to extend the path to v due to other resource limitations, i.e. time or capacity. The visitation resources of a path are therefore consumed either because the nodes have already been visited or because of other resource constraints. The concept of unreachable nodes is attractive because it sharpens the dominance relation. Chabrier [9], Danna and Le Pape [16], and Feillet, Gendreau, and Rousseau [28] proposed other heuristic and exact improvements incorporated in their algorithms for the solution of the elementary shortest path problem. The developments in the solution of the subproblem are interesting because the VRPTW polytope in the elementary approach is embedded in a larger polytope but not a polytope over which $c^T x$ can be minimized in polynomial time. In fact, the authors propose to solve another NP-complete problem instead of the VRPTW. In our view this raises the following research question: Can one find an algorithm for the VRPTW that compute polynomial-time lower bounds that are not dominated by the bounds obtained by the elementary path formulation? We believe that further investigations of the VRPTW polytope $\mathscr{P}_{\text{VRPTW}}$ of Definition 1.2.12 will prove valuable in answering this question.

Kallehauge [50] presented some measurements of the effects of relaxing the condition that paths must be elementary. The measurements followed a suggestion by Natashia Boland, who raised the following research question: What are the gaps between the solutions to (1.69) with elementary routes and solutions to (1.69) with non-elementary routes and 2-cycle elimination for Solomon's data sets? She suggested that a good start would be to look at the non-elementary LP solutions that are produced, and check how many non-elementary paths are assigned non-zero LP values, and measuring the sum of the non-elementary LP variables over the sum of the LP-variables, i.e. the fraction of the non-elementary paths. Kallehauge [51] made these measurements for all Solomon's short-horizon 100 customer problems (excluding C1 problems) and it showed that the problems that remained unsolved have a high fraction of flow on non-elementary paths. In Table 1.1 we present these measurements, which are made in the root node of the branch-and-bound tree before inserting any cuts. NE CUSTOMERS is the number of customers that are visited more than once. NE PATHS/ ALL PATHS is the number of corresponding paths in the LP-solution that contain cycles compared to the total number of paths, i.e. the number of variables greater than zero. NE FLOW/ TOTAL is the sum of the LP-variables that correspond to paths with cycles and the total sum. %NE FLOW shows the percentage of the non-elementary compared to the total flow.

Following a suggestion by Jacques Desrosiers, Kallehauge [51] presented the same measurements as in Table 1.1 after the total flow is integer, i.e. when the number of vehicles is integer. It is also possible to include e.g. subtour inequalities and 2-path inequalities in the LP-model before branching on vehicles. Table 1.2 shows these measurements after subtour inequalities and 2-path inequalities are generated for the LP problem of the root node and we have branched on vehicles. We keep generating these inequalities as long as we only branch on vehicles. The number of times we branch on vehicles is relatively small because the number of vehicles quickly becomes integer. If we compare the column %NE FLOW in Table 1.1 with the equivalent column in Table 1.2 we see that the flow on the non-elementary paths is decreased after inserting cuts and branching on vehicles but it is still significant, i.e. above 20% for all unsolved

	NE CUSTOMERS	NE PATHS/ ALL PATHS	NE FLOW/ TOTAL FLOW	%NE FLOW
R101	0	0 / 26	0 / 19.5	0
R102	0	0 / 18	0 / 18.0	0
R103	8	7 / 39	1.22 / 14.06	8.7
R104*	49	41 / 78	4.18 / 10.16	41.1
R105	0	0 / 59	0 / 14.88	0
R106	8	9 / 74	1.37 / 13.00	10.5
R107	23	29 / 78	2.42 / 10.95	22.1
R108*	34	32 / 73	3.63 / 9.82	37.0
R109	14	14 / 78	1.70 / 12.23	14.0
R110	24	25 / 89	2.86 / 10.96	26.1
R111	17	20 / 91	1.87 / 11.43	16.4
R112*	41	36 / 86	2.79 / 9.49	29.4
RC101	0	0 / 60	0 / 14.58	0
RC102	11	10 / 61	1.88 / 13.51	14.0
RC103	20	24 / 72	3.11 / 10.69	29.0
RC104*	53	48 / 85	4.22 / 9.88	42.7
RC105	2	2 / 35	0.40 / 13.53	3.0
RC106*	29	32 / 89	3.42 / 12.34	27.7
RC107*	31	30 / 74	2.87 / 11.48	25.0
RC108*	40	38 / 65	3.58 / 10.73	33.4

* : unsolved instance by September 2000

Table 1.1: Measuring the effects of the non-elementary relaxation in the root node before any inequalities are generated.

instances.

Table 1.3 shows for each instance the number of k-cycles on the paths with non-zero LP-values in the optimal solution to the LP problem of the root node. k-CYCLES shows the number of cycles by type: 3-cycle/4- cycle/5-cycle/etc. It is characteristic for the instances that remained unsolved that a high fraction of the total number of cycles in the non-elementary LP solution is 3-cycles. At that time we therefore were interested in trying 3-cycle elimination on these instances [51]. Oli B. G. Madsen [68] presented our results at a GERAD seminar in Montreal where Stefan Irnich [44] was visiting at that time. Following this seminar Stefan Irnich [45] generalized the 2-cycle elimination of Houck et al. [43] and Irnich and Villeneuve [46] further extended this work with detailed computational results available in [47]. Indeed the k-cycle elimination gave improvements in the lower bounds and was still computationally attractive as long as k is not too large. Irnich and Villeneuve [46] presented computational results for values of $k = 2,\ldots,5$. Irnich and Villeneuve [46] gave an upper bound on the increase in the number of labels. This bound is $(k-1)!^2$ for $k > 3$ so there are limits to how large values of k should be used. For $k = 2$ the increased number of labels is a factor 2, for $k = 3$ it is a factor 6.

Kallehauge, Larsen, and Madsen [54] presented a stabilized cutting-plane algorithm for the Lagrangian dual problem. The idea is to force the next dual solution of the cutting-plane algorithm to be a priori in a ball or trust-region associated with the given norming. The authors use the max-norm so the master problem is an LP problem with bounds on the dual variables. This is an acceleration of the cutting-plane algorithm of Kelley [56] and Cheney and Goldstein [10]; the original reference for the column generation variant is Dantzig and Wolfe [19]. The trust-region ensures stability of the dual solution from one iteration to the next. Instability refers to the situation where the current iterate is closer (with respect to some norm) to the optimal solution than the next iterate. Kallehauge et al. [54] was motivated

	NE CUSTOMERS	NE PATHS/ ALL PATHS	NE FLOW/ TOTAL FLOW	%NE FLOW
R101	0	0 / 29	0 / 20.00	0
R102	0	0 / 18	0 / 18.00	0
R103	4	4 / 42	1.00 / 15.00	6.7
R104*	45	29 / 81	2.95 / 11.00	26.8
R105	0	0 / 74	0 / 15.00	0
R106	6	7 / 74	1.11 / 13.00	8.5
R107	16	20 / 87	1.69 / 12.00	14.1
R108*	35	35 / 85	2.90 / 10.00	29.0
R109	6	6 / 65	0.85 / 13.00	6.5
R110	23	23 / 92	2.51 / 11.00	22.8
R111	18	19 / 90	1.97 / 12.00	16.4
R112*	40	31 / 91	2.70 / 10.00	27.0
RC101	0	0 / 77	0 / 16.00	0
RC102	12	19 / 97	1.91 / 14.00	13.6
RC103	22	25 / 85	2.97 / 11.00	27.0
RC104*	42	37 / 93	2.93 / 10.00	29.3
RC105	0	0 / 25	0 / 15.00	0
RC106*	28	30 / 99	3.03 / 13.00	23.3
RC107*	37	35 /101	2.85 / 12.00	23.8
RC108*	40	48 / 97	2.95 / 11.00	26.8

* : unsolved instance by September 2000

Table 1.2: Measuring the effects of the non-elementary relaxation after inequalities are generated and branching on vehicles.

by the work on acceleration strategies at the master problem level by Kohl and Madsen [58]. However, using the simple max-norm trust-region method the authors avoided solving the quadratic problems of the version of the bundle algorithm Kohl and Madsen [58] considered and they also avoided the line-searches associated with the bundle algorithm. To obtain feasible integer solutions the cutting-plane algorithm is embedded in the branch, cut, and price framework ABACUS [49]. ABACUS is a C++ class library for solving mixed-integer linear-programs (MILP) by branch, cut, and price. It is interesting to note that the authors' formulation of the MILP presented to ABACUS only involves continuous variables (sic!). Obviously, that is because it is the dual problem (1.68) that is presented to ABACUS. Branching decisions are then based on the dual variables of the master problem, i.e. the path variables of (1.69). The authors also introduce inequalities in the master problem. Because the master problem is stated on the dual variables, subtour and 2-path inequalities are added as columns to this problem. Thienel [83] noted that although ABACUS was designed for linear programming relaxations there were no reasons that the branch and bound algorithm of ABACUS was restricted to this type of relaxation. However, Thienel [83] thought that it would require a generalization of ABACUS to use the system for Lagrangian relaxation. In fact Kallehauge et al. [54] showed that by remaining within the context of linear programming when solving the Lagrangian dual problem it is already possible to embed Lagrangian relaxation in this system. The trust-region method of Kallehauge et al. [54] can be used to solve any Lagrangian dual problem associated with a hard optimization problem and its implementation in ABACUS would allow the developer to concentrate on the problem specific parts, i.e. the cutting plane and the column generation, the branching rules, and the primal heuristics.

Recently Jepsen, Spoorendonk, and Petersen [48] introduced 3-customer clique inequalities that are valid for the set-partitioning polytope of (1.69). These inequalities change the structure of the shortest

	k-CYCLES
R101	0
R102	0
R103	4/1/1/1/1/0/0/1
R104*	29/7/11/10/6/6/8/4/7
R105	0
R106	1/0/1/1/3/3
R107	7/5/1/9/6/5/3/2
R108*	26/2/4/5/2/8/6/8/5
R109	19/0/1
R110	24/4/2/1/2/0/1
R111	10/4/3/0/2/1/2/1
R112*	54/7/14/8/0/0/0/1
RC101	0
RC102	5/2/2/1/2/2/3/1
RC103	18/5/6/3/2/1/2
RC104*	55/9/8/9/5/9/6/7/4
RC105	2
RC106*	36/8
RC107*	38/3/6/3/3/2
RC108*	55/7/3/1/5/1/3/3

* : unsolved instance by September 2000

Table 1.3: Number of *k*-cycles (3-cycles/4-cycles/5-cycles/etc.) on paths with non-zero LP-values in root node.

path subproblem and the authors describe how the dominance relation of the subproblem is modified to incorporate these clique inequalities. This is the first example of strengthening the path formulation by introducing inequalities defined directly with respect to the path variables of the master problem.

1.7 Conclusions

In this paper we have reviewed four different formulations of the VRPTW and the exact approaches associated with them. We have identified and organized a total of 24 references on the VRPTW relative to four seminal papers on formulations of the TSP: arc formulation, arc-node formulation, spanning tree formulation, and path formulation. Out of these 24 references are 20 references related to the path formulation of the VRPTW. The polyhedral approach of the arc formulation is in our opinion promising and relatively little research has been conducted along these lines compared to the decomposition approach of the path formulation. Furthermore, the spanning tree formulation of the VRPTW has not been the subject of the same amount of research as the path formulation and the extension of the shortest spanning tree subproblem to time windows has not been considered in the literature. In our view it is at this point therefore an open question whether the spanning tree formulation described in this paper is effective compared to the path formulation.

The exact approaches based on the path formulation has been very successful and the most important contributions of the research on the VRPTW lies in this area. The developments in the solution of the subproblem are interesting because the VRPTW polytope in the elementary path formulation approach is embedded in a larger polytope but not a polytope over which the objective function can be minimized in polynomial time. In fact, the authors propose to solve another NP-complete problem instead of the

VRPTW, i.e. a resource-constrained elementary shortest path problem. In our view this raises the question of one can find an algorithm for the VRPTW that compute polynomial-time lower bounds that are not dominated by the bounds obtained by the elementary path formulation. We believe that further investigations of the VRPTW polytope will prove valuable in answering this question. The developments in the solution of the (dual) master problem associated with the path formulation are also very interesting. There are at least three important developments. First, introduction of strong valid inequalities for the VRPTW polytope in the master problem, e.g. generalized subtour inequalities. Second, development of acceleration techniques that addresses the instability issues with the cutting-plane algorithm for convex minimization or equivalently the column generation algorithm of the Dantzig-Wolfe decomposition. Third, and more recently, strong valid inequalities have been introduced for the set-partitioning polytope and thereby strengthening the lower bounds provided by this relaxation. The inequalities can also be used from a dual point of view in the Lagrangian dual problem.

It is clear that 'A new optimization algorithm for the vehicle routing problem with time windows' (Desrochers, Desrosiers, and Solomon, 1992) was a very substantial achievement. It is important both for introducing the path formulation and the column generation algorithm to the VRPTW and for the future developments it inspired. It is remarkable how much of the scope and methodology of combinatorial optimization has been applied in the attack on the VRPTW. The importance of the research described in this paper comes not from the number of applications where the mathematical model of the VRPTW precisely fits, but from the fact that the vehicle routing problem with time windows is typical of other resource-constrained problems in combinatorial optimization.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36:69–79, 2000. [7]

[2] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming Series A*, 90:475–506, 2001. [9]

[3] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995. [7]

[4] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002. [4, 9]

[5] J. E. Beasley. Lagrangean relaxation. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Advanced topics in computer science, pages 243–303. McGraw-Hill, London, 1995. [13]

[6] J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. [20]

[7] M. Bellmore and S. Hong. Transformation of multisalesmen problem to the standard traveling salesman problem. *Journal of the Association for Computing Machinery*, 21:500–504, 1974. [6]

[8] J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 85–108. SIAM, Philadelphia, 2002. [2, 17]

[9] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. To appear in Computers & Operations Research. [4, 20]

[10] E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959. [21]

[11] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981. [13, 18]

[12] N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11:145–164, 1981. [18]

[13] COIN/BCP User's Manual. 2001. [3]

[14] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, Rice University, 1999. [4, 19]

[15] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 157–193. SIAM, Philadelphia, 2002. [2, 4]

[16] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column generation*, GERAD 25th Anniversary Series, pages 99–129. Springer, New York, 2005. [4, 20]

[17] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problems. *Operations Research*, 2:393–410, 1954. [2, 4, 7, 8]

[18] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959. [1, 2, 3]

[19] G. B. Dantzig and P. Wolfe. A decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960. [21]

[20] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 225–242. SIAM, Philadelphia, 2002. [2]

[21] M. Desrochers. An algorithm for the shortest path problem with resource constraints. Technical report, GERAD, 1988. [14, 18]

[22] M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991. [8]

[23] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992. [4, 14, 18, 20, 24]

[24] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14:545–565, 1984. [18]

[25] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, The Netherlands, 1995. [18]

[26] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994. [14]

[27] H. Everett, III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963. [3]

[28] D. Feillet, M. Gendreau, and L.-M. Rousseau. New refinements for the solution of vehicle routing problems with branch and price. Submitted. [4, 20]

[29] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44:216–229, 2004. [4, 20]

[30] M. L. Fisher. Optimal solution of vehicle routing problems using minimum K-trees. *Operations Research*, 42:626–642, 1994. [13]

[31] M. L. Fisher, K. O. Jörnsten, and O. B. G. Madsen. Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45:488–492, 1997. [4, 12, 13, 19]

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979. [6]

[33] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974. [13]

[34] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem – part II. *Operations Research*, 11:863–888, 1963. [13]

[35] M. Grötschel and M. W. Padberg. Polyhedral theory. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization*, pages 251–305. Wiley, New York, 1985. [7]

[36] M. Guignard and S. Kim. Lagrangean decomposition: a model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39:215–228, 1987. [18]

[37] K. Halse. *Modeling and solving complex vehicle routing problems*. PhD thesis, Department of Mathematical Statistics and Operations Research, Technical University of Denmark, 1992. [4, 18]

[38] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970. [3, 4, 12, 13]

[39] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971. [3, 4, 13, 17]

[40] M. Held, P. Wolfe, and P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974. [13]

[41] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I-II*. Grundlehren der matematischen Wissenschaften 304-305. Springer-Verlag, Berlin Heidelberg, 1993. [3, 13, 19]

[42] A. J. Hoffman and P. Wolfe. History. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization*, pages 1–15. Wiley, New York, 1985. [2]

[43] D. J. Houck, Jr., J. C. Picard, M. Queyranne, and R. R. Vemuganti. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *OPSEARCH*, 17: 93–109, 1980. [4, 14, 17, 18, 21]

[44] S. Irnich. The mixed direct and hub flight problem, a branch-and-price approach for simultaneous network design and scheduling, September, 2000. GERAD seminar. [21]

[45] S. Irnich. The shortest path problem with k-cycle elimination ($k \geqslant 3$): Improving a branch and price algorithm for the VRPTW. Technical report, Lehr- und Forschungsgebiet Unternehmensforschung, Rheinisch-Westfälische Technische Hochschule, November 22, 2000. [21]

[46] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. To appear in INFORMS Journal on Computing, . [4, 14, 21]

[47] S. Irnich and D. Villeneuve. Online supplement for the shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$, .
`http://joc.pubs.informs.org/OnlineSupplements.html`. [21]

[48] M. Jepsen, S. Spoorendonk, and B. Petersen. A branch-and-cut-and-price framework for VRP applied on CVRP and VRPTW. Master's thesis, Department of Computer Science, University of Copenhagen, 2005. [4, 22]

[49] M. Jünger and S. Thienel. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software: Practice and Experience*, 30:1325–1352, 2000. [3, 22]

[50] B. Kallehauge. A hybrid optimal method to the vehicle routing problem with time windows. In *ROUTE 2000 - International workshop on vehicle routing*, Skodsborg, Denmark, August 16-19, 2000. Presentation available at `http://www1.ctt.dtu.dk/ROUTE2003/route2000/presentations/presentation_brian.pdf`. [20]

[51] B. Kallehauge. Ideas for solving VRPTW, September 21, 2000. Private communication to Jacques Desrosiers, Oli B.G. Madsen and Jesper Larsen. [20, 21]

[52] B. Kallehauge, N. Boland, and O. B. G. Madsen. Path inequalities for the vehicle routing problem with time windows. Submitted. [4, 7, 9]

[53] B. Kallehauge, J. Larsen, O. B. G. Madsen, and M. M. Solomon. Vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column generation*, GERAD 25th Anniversary Series, pages 67–98. Springer, New York, 2005. [4, 17]

[54] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33:1464–1487, 2006. [4, 13, 21, 22]

[55] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the Association for Computing Machinery*, 43:601–640, 1996. [19]

[56] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of SIAM*, 8:703–712, 1960. [21]

[57] N. Kohl. *Exact methods for time constrained routing and related scheduling problems*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1995. [18, 19]

[58] N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean relaxation. *Operations Research*, 45:395–406, 1997. [4, 19, 22]

[59] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999. [4, 7, 19]

[60] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987. [4, 18]

[61] R. V. Kulkarni and P. R. Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20:58–67, 1985. [8]

[62] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1050–1073, 1985. [3, 7, 13]

[63] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1999. [4, 19]

[64] J. Larsen. Refinements of the column generation process for the vehicle routing problem with time windows. *Journal of Systems Science and Systems Engineering*, 13:326–341, 2004. [4, 19]

[65] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. Wiley, New York, 1985. [2]

[66] C. Lemaréchal. A view of line-searches. In A. Auslender, W. Oettli, and J. Stoer, editors, *Optimization and Optimal Control*, volume 30 of *Lecture Notes in Control and information Sciences*, pages 59–78. Springer, Berlin Heidelberg, 1981. [19]

[67] C. Lemaréchal, J. J. Strodiot, and A. Bihain. On a bundle algorithm for nonsmooth optimization. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 4*, pages 245–282. Academic Press, New York, 1981. [19]

[68] O. B. G. Madsen. Vehicle routing problems with time windows - some recent developments, September, 2000. GERAD seminar. [21]

[69] V. Mak and A. Ernst. New cutting-planes for the time and/or precedence constrained ATSP and directed VRP. Submitted. [4, 7]

[70] V. H. Mak. *On the Asymmetric Travelling Salesman Problem with Replenishment Arcs*. PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, 2001. [7]

[71] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960. [4, 6, 8]

[72] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 49–78. SIAM, Philadelphia, 2002. [2, 7]

[73] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988. [3]

[74] W. Orchard-Hays. *Advanced Linear-Programming Computing Techniques*. McGraw-Hill, New York, 1968. [17]

[75] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991. [3]

[76] C. H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8:217–230, 1978. [10]

[77] J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26:86–110, 1978. [17]

[78] T. Ralphs and M. Guzelsoy. The SYMPHONY callable library for mixed integer programming. Technical report, Department of Industrial and Systems Engineering, Leheigh University, 2004. [3]

[79] J. L. Rich. *A Computational Study of Vehicle Routing Applications*. PhD thesis, Rice University, 1999. [19]

[80] A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Algorithms and Combinatorics 24. Springer, Berlin Heidelberg, 2003. [3, 14]

[81] J. F. Shapiro. *Mathematical Programming: Structures and Algorithms*. Wiley, New York, 1979. [3]

[82] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987. [7, 18]

[83] S. Thienel. *ABACUS A Branch-And-CUt System*. PhD thesis, Mathematisch-Naturwissenschaftlichen Fakultät, Universität zu Köln, 1995. [3, 22]

[84] P. Toth and D. Vigo. An exact algorithm for the capacitated shortest spanning arborescence. *Annals of Operations Research*, 61:121–141, 1995. [13]

[85] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2002. [2]

[86] P. Toth and D. Vigo. Branch-and-bound algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 29–51. SIAM, Philadelphia, 2002. [2]

[87] P. Toth and D. Vigo. VRP with backhauls. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 195–224. SIAM, Philadelphia, 2002. [2]

Chapter 2

Lagrangian duality applied to the vehicle routing problem with time windows

Brian Kallehauge
Centre for Traffic and Transport, Technical University of Denmark

Jesper Larsen
Informatics and Mathematical Modelling, Technical University of Denmark

Oli B. G. Madsen
Centre for Traffic and Transport, Technical University of Denmark

Abstract

This paper considers the vehicle routing problem with time windows, where the service of each customer must start within a specified time interval. We consider the Lagrangian relaxation of the constraint set requiring that each customer must be served by exactly one vehicle yielding a constrained shortest path subproblem. We present a stabilized cutting-plane algorithm within the framework of linear programming for solving the associated Lagrangian dual problem. This algorithm creates easier constrained shortest path subproblems because less negative cycles are introduced and it leads to faster multiplier convergence due to a stabilization of the dual variables. We have embedded the stabilized cutting-plane algorithm in a branch-and-bound search and introduce strong valid inequalities at the master problem level by Lagrangian relaxation. The result is a Lagrangian branch-and-cut-and-price (LBCP) algorithm for the VRPTW. Making use of this acceleration strategy at the master problem level gives a significant speed-up compared to algorithms in the literature based on traditional column generation. We have solved two test problems introduced in 2001 by Gehring and Homberger with 400 and 1000 customers respectively, which to date are the largest problems ever solved to optimality. We have implemented the LBCP algorithm using the ABACUS open-source framework for solving mixed-integer linear-programs by branch, cut, and price.

2.1 Introduction

This paper considers the following variant of the vehicle routing problem with time windows (VRPTW): Consider a directed graph $D = (V,A)$ on $n+2$ nodes. The node 0 corresponds to a depot and the node $n+1$ is a copy of node 0. Let the set K correspond to a fleet of identical vehicles available at the depot source node 0 and each with a capacity q. Nodes $i \in N = \{1,\ldots,n\}$ correspond to customers to be served by a vehicle $k \in K$. With each customer node $i \in N$ we associate a demand $0 \leqslant d_i \leqslant q$, a service time $st_i \geqslant 0$, a release time $a_i \geqslant 0$ and a due time $b_i \geqslant a_i$. The release time a_i and the due time b_i is the earliest possible and the latest possible starting time respectively for serving customer $i \in N$. We refer to the time interval $[a_i,b_i]$ as the time window of customer $i \in N$ and $b_i - a_i$ as the width of the time window. We assume that $d_0 = st_0 = 0$ and w.l.o.g that $[a_0,b_0] = [0,\infty)$. Let $A = \{(0,j) \mid j \in N\} \cup \{(i,j) \mid i \in N, j \in N, a_i + t_{ij} \leqslant b_j \text{ and } d_i + d_j \leqslant q\} \cup \{(i,n+1) \mid i \in N\}$. With each arc $(i,j) \in A$, we associate an arc cost $c_{ij} \geqslant 0$ and an arc travel time $t_{ij} > 0$, which include any service time st_i at node i. We assume that demands, service times, release times, due times, costs, travel times, and vehicle capacity are integer values. It is also assumed that the triangle inequality on the costs and travel times is satisfied, i.e. $c_{ij} \leqslant c_{ih} + c_{hj}$ and $t_{ij} \leqslant t_{ih} + t_{hj}$, for all $(i,j) \in A$. Given a vehicle $k \in K$ a path p_k consists of the arc set $\{(0,v_1)\} \cup \{(v_i,v_{i+1}) \mid i = 1,\ldots,h-1\} \cup \{(v_h,n+1)\}$, where $v_i \neq v_j$, for $i \neq j$. Such a path represents the trip of one vehicle $k \in K$ leaving the depot source node 0, collecting the demand d_i of the nodes v_i, $i = 1,\ldots,h$, and going back to the depot sink node $n+1$. We consider the case where a vehicle $k \in K$ waits if it arrives at a node $i \in N$ before the release time a_i, i.e. given a path p_k let s_{v_i} denote the earliest start of service of node v_i, $i = 1,\ldots,h$, defined as follows.

$$
\begin{aligned}
& s_{v_1} = a_{v_1}, \\
& s_{v_i} = \max\{s_{v_{i-1}} + t_{v_{i-1},v_i}, a_{v_i}\} \quad \text{for } i = 2,\ldots,h.
\end{aligned}
$$

This gives a waiting time $w_{v_i} = \max\{0, a_{v_i} - (s_{v_{i-1}} + t_{v_{i-1}v_i})\}$, which is positive when a vehicle arrives at a node before its release time. We allow vehicles to remain at the depot. This is modeled by introducing the arc $(0,n+1) \in A$, with $c_{0,n+1} = t_{0,n+1} = 0$ and thereby the "empty path" $p_k = \{(0,n+1)\}$, which represents an unused vehicle $k \in K$. We assume that the number of vehicles used is free. This is modeled by setting $|K| = n$ or another valid upper bound. A path p_k is called feasible with respect to a vehicle $k \in K$ if the total demand $\sum_{i=1}^{h} d_i$ of the nodes in p_k does not exceed the vehicle capacity q and each node v_i is visited within its time window, i.e. $a_{v_i} \leqslant s_{v_i} \leqslant b_{v_i}$, for $i = 1,\ldots,h$. Given a vehicle $k \in K$ we denote the set of all feasible paths P_k.

For a subset F of A, $D(F)$ denotes the subdigraph $(V(F),F)$ induced by F, where $V(F)$ is the set of nodes incident to at least one edge of F.

The cost of a path is the sum of the costs of arcs used in the path, i.e. $\sum_{(i,j)\in p_k} c_{ij}$. An m-path is the union of m paths $p_1,\ldots,p_m$, such that each node $i \in N$ belongs to exactly one set $N(p_k)$, $k = 1,\ldots,m$, and $|K| - m$ paths representing unused vehicles, i.e. $m \leqslant |K|$ denotes the number of vehicles used to serve the customers $i \in N$. An m-path is feasible if it consists only of feasible paths. The cost of an m-path is $\sum_{k=1}^{m} \sum_{(i,j)\in p_k} c_{ij}$. The problem is to find a minimum cost feasible m-path. We would like to note that our presentation of the VRPTW problem is inspired by Ascheuer et al. [1] and Naddef and Rinaldi [30].

The VRPTW reduces to the VRP if $st_i = 0$, $a_i = 0$, and $b_i = \infty$ for every $i \in N$. Therefore the VRPTW is NP-hard. Indeed, it is strongly NP-complete to find a feasible solution for the VRPTW with a fixed number of vehicles [33].

The VRPTW variant we are considering has been attacked by various methods of integer programming. In Cordeau et al. [6] a review of the research until 2000 on exact methods for this variant is given. The most successful methods in this period are based on the path formulation of Desrochers et al. [10], involving binary variables associated with feasible paths in the underlying time and capacity constrained digraph. Kohl and Madsen [23] proposed an equivalent method based on Lagrangian relaxation. Both

decomposition methods split the constraints into the same two sets, yielding the same constrained shortest path subproblem and hence the same lower bound on the VRPTW. The subproblem is NP-hard in the strong sense [11] and a non-elementary state-space relaxation allowing cycles of length greater than two is solved. In the Dantzig-Wolfe decomposition method the master problem is a set-partitioning problem. The linear programming relaxation of the master problem is solved using the column generation algorithm. In the Lagrangian relaxation method the master problem can be formulated as the maximization of a concave nonsmooth function, piecewise affine in the Lagrangian multipliers [14]. This problem is denoted the Lagrangian dual problem. Kohl and Madsen [23] solved the Lagrangian dual problem using a combination of a subgradient algorithm and a bundle algorithm. A bundle algorithm can be viewed as an acceleration of the column generation algorithm [17, Chapter XII and XV]. Kohl et al. [24] proposed a strengthening to the path formulation involving constraints for general subtour elimination and Cook and Rich [5] extended this approach. This strengthened path formulation is exponential in size since it has exponentially many variables as well as constraints and a column and cut generation approach was used for solving the problem. Recently Irnich and Villeneuve [19] proposed a strengthening to the bound provided by the path formulation involving elimination of cycles of length greater than two in the subproblem. Algorithms for the elementary shortest path subproblem have also recently been proposed in Feillet et al. [13] and Chabrier [2].

The research in this paper was motivated by the work on acceleration strategies at the master problem level by Kohl and Madsen [23]. However we have chosen to remain within the context of linear programming and the focus is therefore on accelerating the classical cutting-plane algorithm of Kelley [21] and Cheney and Goldstein [3]; the original reference for the column generation variant is Dantzig and Wolfe [7]. In a different application du Merle et al. [12] proposed ways to accelerate the cutting-plane and column generation algorithm. The algorithm we present creates easier constrained shortest path subproblems because less negative cycles are introduced and it leads to faster multiplier convergence due to a stabilization of the dual variables.

The main contributions of this paper are:

- Developing a stabilized cutting-plane algorithm within the framework of linear programming for solving the Lagrangian dual problem associated with the Lagrangian relaxation of the assignment constraints of the VRPTW.
- Embedding the stabilized cutting-plane algorithm in a branch-and-bound algorithm and introducing strong valid inequalities at the master problem level by Lagrangian relaxation. The result is a Lagrangian branch-and-cut-and-price (LBCP) algorithm for the VRPTW.
- Decreasing the solution times significantly for a large number of Solomon VRPTW problems compared with a traditional column generation based algorithm.

The paper is organized as follows. In Section 2, we give an integer linear programming (ILP) formulation of the VRPTW. Section 3 presents a Lagrangian relaxation of the VRPTW and shows that the Lagrangian dual problem can be formulated as a linear programming problem. Section 4 briefly describes the algorithm used for solving the Lagrangian problem, which can be split into a subproblem for each vehicle. In Section 5, we present a stabilized cutting-plane algorithm for solving the Lagrangian dual problem and in Section 6 we present the LBCP algorithm used for finding integer solutions. Section 7 presents computational results. We compare the numerical performance of the stabilized cutting-plane algorithm to a traditional column generation algorithm and we test the LBCP algorithm on the Solomon [34] problems and on a number of larger problems created by Gehring and Homberger [15]. Finally, Section 8 presents our conclusions.

2.2 An ILP formulation of the VRPTW

The model involves binary arc variables x_{ijk} as well as integer node variables s_{ik} for each vehicle. For a given vehicle $k \in K$ the binary arc variables x_{ijk} for each arc $(i,j) \in A$ are defined as follows.

$$x_{ijk} = \begin{cases} 1 & (i,j) \in A \text{ is used by vehicle } k \text{ in the } m\text{-path}, \\ 0 & \text{otherwise.} \end{cases}$$

The decision variables s_{ik} are defined for each node $i \in N$ and each vehicle $k \in K$ and represent the time vehicle k starts to service customer i. In case the given vehicle k does not service customer i s_{ik} does not mean anything. The VRPTW is formulated as the following integer linear program:

$$z_{\text{VRPTW}} = \text{minimize} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{2.1}$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \quad \forall i \in N, \tag{2.2}$$

$$\sum_{j \in V} x_{0jk} = 1 \quad \forall k \in K, \tag{2.3}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad \forall h \in N, \forall k \in K, \tag{2.4}$$

$$\sum_{i \in V} x_{i,n+1,k} = 1 \quad \forall k \in K, \tag{2.5}$$

$$\sum_{i \in N} d_i \sum_{j \in V} x_{ijk} \leqslant q \quad \forall k \in K, \tag{2.6}$$

$$s_{ik} + t_{ij} - L_{ij}(1 - x_{ijk}) \leqslant s_{jk} \quad \forall i,j \in N, \forall k \in K, \tag{2.7}$$

$$a_i \leqslant s_{ik} \leqslant b_i \quad \forall i \in N, \forall k \in K, \tag{2.8}$$

$$x_{ijk} \in \{0,1\} \quad \forall (i,j) \in A, \forall k \in K, \tag{2.9}$$

$$s_{ik} \in \mathbb{Z}_+ \quad \forall i \in N, \forall k \in K, \tag{2.10}$$

where $L_{ij} = b_i - a_j$ and $\mathbb{Z}_+$ denotes the set of nonnegative integers.

The objective function (2.1) expresses the total cost. Equalities (2.2) are the assignment constraints requiring each customer to be served by exactly one vehicle. Equalities (2.3)-(2.5) are the out-degree, flow balance, and in-degree constraints forcing the solution to consist of a set of paths. Inequalities (2.6) are the capacity constraints and inequalities (2.7) and (2.8) are used to model the time window restrictions. Finally, (2.9) and (2.10) are the binary and integer constraints.

2.3 A Lagrangian relaxation of the VRPTW

We consider the Lagrangian relaxation of the VRPTW with respect to the constraints (2.2), by introducing a vector of Lagrangian multipliers $\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n$, where λ_i is associated with the ith constraint in (2.2):

$$z_{\text{D}}(\lambda) = \underset{\substack{\text{subject to} \\ \text{(2.3)-(2.10)}}}{\text{minimize}} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} - \sum_{i \in N} \lambda_i \left(\sum_{k \in K} \sum_{j \in V} x_{ijk} - 1 \right). \tag{2.11}$$

We call (2.11) the Lagrangian problem. The minimal value in the Lagrangian problem (2.11) is called the dual function and is denoted z_{D}. The set of feasible solutions to the Lagrangian problem (2.11)

defined by (2.3)-(2.10) is denoted P_λ. P_λ splits into $|K|$ disjoint subsets, i.e. $P_\lambda = P_1 \times P_2 \times \cdots \times P_{|K|} = \{(p_1,\ldots,p_{|K|}) \mid p_1 \in P_1,\ldots,p_{|K|} \in P_{|K|}\}$, where each P_k is defined by (2.3)-(2.10) for a given $k \in K$. The Lagrangian problem (2.11) therefore splits into $|K|$ "simpler" subproblems, one for each vehicle, and the kth subproblem is then the integer linear program:

$$z_k(\lambda) = \text{minimize} \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \tag{2.12}$$

subject to

$$\sum_{j\in V} x_{0j} = 1, \tag{2.13}$$

$$\sum_{i\in V} x_{ih} - \sum_{j\in V} x_{hj} = 0 \quad \forall h \in N, \tag{2.14}$$

$$\sum_{i\in V} x_{i,n+1} = 1, \tag{2.15}$$

$$\sum_{i\in N} d_i \sum_{j\in V} x_{ij} \leqslant q, \tag{2.16}$$

$$s_i + t_{ij} - L_{ij}(1 - x_{ij}) \leqslant s_j \quad \forall i,j \in N, \tag{2.17}$$

$$a_i \leqslant s_i \leqslant b_i \quad \forall i \in N, \tag{2.18}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V, \tag{2.19}$$

$$s_i \in \mathbb{Z}_+ \quad \forall i \in N, \tag{2.20}$$

where $\tilde{c}_{ij} = c_{ij} - \lambda_i$ for $i \in N, j \in V$, $\tilde{c}_{ij} = c_{ij}$ otherwise, and admitting that x_{ij} stands for x_{ijk}, but with k fixed.

Constraints (2.13)-(2.15) force the solution $p_k \in P_k$ to represent a path starting in the depot source node 0 and ending in the depot sink node $n+1$, whereas the capacity constraints (2.16) and the time constraints (2.17) and (2.18) forbid infeasible paths to be part of the solution. This subproblem (2.12)-(2.20) is an elementary shortest path problem with time windows and capacity constraints (ESPPTWCC), where each node can participate at most once in the path $p_k \in P_k$. Note that while c_{ij}, $(i,j) \in A$, is a non-negative integer, $\tilde{c}_{ij}$, $i \in N, j \in V$, may be any real number.

The subproblems (2.12)-(2.20) for $k \in K$ are identical, which means the Lagrangian problem (2.11) is expressed as:

$$z_{\mathrm{D}}(\lambda) = |K|\Big(\underset{\substack{\text{subject to}\\ \text{(2.13)-(2.20)}}}{\text{minimize}} \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij}\Big) + \sum_{i\in N} \lambda_i \tag{2.21}$$

Since the subsets P_k for $k \in K$ are identical $P_\lambda = P^{|K|}$, where P denotes the set of feasible solutions to the ESPPTWCC subproblem (2.12)-(2.20) for any vehicle $k \in K$. Let $z(\lambda)$ denote the minimal solution value for $\lambda \in \mathbb{R}^n$. We describe each path $p \in P$ with the integer variables x_{ijp}, $(i,j) \in A$. Given a path $p \in P$ let c_p be the cost of path $p \in P$ and let a_{ip} be the number of times customer i is served on path p:

$$c_p = \sum_{(i,j)\in A} c_{ij} x_{ijp}, \text{ for } p = 1,\ldots,|P|,$$

$$a_{ip} = \sum_{j\in V} x_{ijp}, \text{ for } i = 1,\ldots,|N|,\ p = 1,\ldots,|P|.$$

Since P is finite, we can consider $z(\lambda)$ to be determined by minimization over the set P of constrained shortest paths. Then (2.11) is expressed as:

$$z_{\mathrm{D}}(\lambda) = |K|\Big(\underset{p\in P}{\text{minimize}}\, c_p - \sum_{i\in N} a_{ip}\lambda_i\Big) + \sum_{i\in N} \lambda_i \tag{2.22}$$

The fact that $z_{\mathrm{D}}(\lambda) \leqslant z_{\mathrm{VRPTW}}$, for $\lambda \in \mathbb{R}^n$, provides us with lower bounds in a branch-and-bound algorithm for the VRPTW. Clearly, we wish to find the best lower bound by solving the Lagrangian dual problem:

$$
\begin{aligned}
(2.23) \qquad z_{\mathrm{LD}} &= \underset{\lambda \in \mathbb{R}^n}{\text{maximize}}\, z_{\mathrm{D}}(\lambda) \\
&= \underset{\lambda \in \mathbb{R}^n}{\text{maximize}}\, |K| \Big(\underset{p \in P}{\text{minimize}}\, c_p - \sum_{i \in N} a_{ip} \lambda_i \Big) + \sum_{i \in N} \lambda_i .
\end{aligned}
$$

Since P is finite it allows us to express (2.23) as the following linear program with many constraints or rows:

$$
\begin{aligned}
(2.24) \qquad & z_{\mathrm{LD}} = \underset{\substack{\lambda \in \mathbb{R}^n \\ \theta \in \mathbb{R}}}{\text{maximize}}\, |K| \theta + \sum_{i \in N} \lambda_i \\
& \text{subject to} \\
& \theta \leqslant c_p - \sum_{i \in N} a_{ip} \lambda_i \quad \text{for all } p \in P.
\end{aligned}
$$

The LP dual of (2.24) is a linear program with many variables or columns:

$$
\begin{aligned}
(2.25) \qquad & z_{\mathrm{LD}} = \text{minimize} \sum_{p \in P} c_p y_p \\
& \text{subject to} \\
& \sum_{p \in P} a_{ip} y_p = 1 \quad \text{for all } i \in N, \\
& \sum_{p \in P} y_p = |K|, \\
& y_p \geqslant 0 \quad \text{for all } p \in P.
\end{aligned}
$$

Problem (2.25) with y_p required to be integral is equivalent to the original VRPTW formulation (2.1)-(2.10). Problem (2.25) is the LP relaxation of the Dantzig-Wolfe decomposition obtained when any solution to the VRPTW is expressed as a non-negative convex combination of constrained paths [6]. The method of Desrochers et al. [10] can be characterized as column generation on the problem (2.25) and their formulation of the VRPTW can therefore be viewed as a Dantzig-Wolfe decomposition of the formulation (2.1)-(2.10). Note that $\{(0, n+1)\} \in P$, $c_{0,n+1} = 0$, and requiring a fixed number of vehicles $|K|$ is equivalent with having a free number of vehicles in (2.25).

Let $\mathbb{R}^A$ be the set of real vectors whose components are indexed by A. Let $x = (x_1, \ldots, x_{|K|})$, where $x_k \in \mathbb{R}^A$. Now consider the Lagrangian function or *Lagrangian*:

$$
\begin{aligned}
(2.26) \qquad L(\lambda, x) &= \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} + \sum_{i \in N} \lambda_i \Big(1 - \sum_{k \in K} \sum_{j \in V} x_{ijk} \Big) \\
&= |K| \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{i \in N} \lambda_i \Big(1 - |K| \sum_{j \in V} x_{ij} \Big),
\end{aligned}
$$

which shows that the Lagrangian is a function of say $x_1 \in \mathbb{R}^A$ only. For a fixed solution $x_p = x_{1p} \in P$, the Lagrangian is an affine function in λ:

$$
(2.27) \qquad L(\lambda, x_p) = \langle s_p, \lambda \rangle + |K| c_p,
$$

where $\langle \cdot, \cdot \rangle$ denotes the ordinary dot-product and $s_p = (s_{1p}, \ldots, s_{np}) \in \mathbb{R}^n$, where:

$$
(2.28) \qquad s_{ip} = 1 - |K| \sum_{j \in V} x_{ijp}.
$$

The dual function becomes:

$$z_{\mathrm{D}}(\lambda) = \underset{1 \leqslant p \leqslant |P|}{\text{minimize}} \langle s_p, \lambda \rangle + |K| c_p, \tag{2.29}$$

which shows that the dual function z_{D} (2.11) is the minimum of a finite number of affine functions and is therefore piecewise affine and concave [32].

The dual function is non-differentiable or non-smooth at any point $\lambda \in \mathbb{R}^n$ where the solution of the Lagrangian problem (2.11) is not unique. This corresponds to the subproblem (2.12) - (2.20) having several shortest path solutions.

The subdifferential of the dual function at λ is given by the convex hull of the gradients of the Lagrangian functions that give the minimal value:

$$\begin{aligned} \partial z_{\mathrm{D}}(\lambda) &= \mathrm{conv}\{\nabla L(\lambda, x_p) \mid L(\lambda, x_p) = z_{\mathrm{D}}(\lambda)\} \\ &= \mathrm{conv}\{s_p \mid \langle s_p, \lambda \rangle + |K| c_p = z_{\mathrm{D}}(\lambda)\} \end{aligned} \tag{2.30}$$

and the elements of the subdifferential are called subgradients.

However, any suboptimal solution of the Lagrangian problem may also be used. Neame et al. [31] presents an outer approximation of the subdifferential in connection with Lagrangian duality. First define an index set:

$$P_E(\lambda) = \{p \mid \langle s_p, \lambda \rangle + |K| c_p \leqslant z_{\mathrm{D}}(\lambda) + E, E \geqslant 0\}, \tag{2.31}$$

which is the set of solutions to the Lagrangian problem where the solution values are less or equal to the dual function value plus a positive constant. Then the outer approximation to the subdifferential is:

$$\partial_E z_{\mathrm{D}}(\lambda) = \mathrm{conv}\{s_p \mid p \in P_E(\lambda)\}. \tag{2.32}$$

2.4 Solving the Lagrangian problem

We compute a solution to the Lagrangian problem (2.11) by solving an ESPPTWCC. However, Dror [11] proves that the ESPPTWCC is NP-hard in the strong sense, which justifies the approach of considering the non-elementary state-space relaxation SPPTWCC in the papers by Desrochers et al. [10], Kohl and Madsen [23] and Kohl et al. [24] on decompositions methods for the VRPTW.

In the SPPTWCC non-simple or non-elementary paths are allowed, i.e. paths including cycles, and for this problem pseudo-polynomial dynamic programming algorithms are known [9, 8].

In order to tighten the state-space relaxation and increase the lower bound provided by the SPPTWCC subproblem a standard procedure is to use the 2-cycle elimination scheme proposed by Houck et al. [18], in which cycles of the form $i - j - i$ are eliminated.

In this paper we also consider the non-elementary relaxation SPPTWCC of the subproblem (2.12)-(2.20). The SPPTWCC is solved using a label setting algorithm with 2-cycle elimination developed by Larsen [25]. The algorithm not only computes the shortest path solution, but may also return a number of non-dominated paths with negative reduced cost in the sink depot.

2.5 Solving the Lagrangian dual problem

We state our Lagrangian dual problem (2.23) as the following maximin problem:

$$z_{\mathrm{LD}} = \underset{\lambda \in \mathbb{R}^n}{\text{maximize}} \ \underset{1 \leqslant p \leqslant |P|}{\text{minimize}} L(\lambda, x_p) \tag{2.33}$$

which is equivalent to solving the linear program:

$$
\begin{aligned}
(2.34) \qquad & z_{\mathrm{LD}} = \underset{\lambda \in \mathbb{R}^n}{\text{maximize}}\, \theta \\
& \text{subject to} \\
& \theta \leqslant L(\lambda, x_p) \quad p = 1 \ldots, |P|,
\end{aligned}
$$

Now, since the Lagrangian function (2.27) is an affine function in λ for a fixed solution x_p, we have:

$$
(2.35) \qquad L(\lambda, x_p) = z_{\mathrm{D}}(\lambda_p) + \langle s_p, \lambda - \lambda_p \rangle
$$

so the dual problem (2.34) can be written using only dual objects, i.e. dual function values and subgradients:

$$
\begin{aligned}
(2.36) \qquad & z_{\mathrm{LD}} = \text{maximize}\, \theta \\
& \text{subject to} \\
& \theta \leqslant z_{\mathrm{D}}(\lambda_p) + \langle s_p, \lambda - \lambda_p \rangle \quad \text{for } p = 1, \ldots, |P|.
\end{aligned}
$$

The idea of the cutting-plane algorithm is to accumulate the constraints one after the other in (2.36) [17].

Now we will describe the rules that our algorithm follows in order to generate the constraints of (2.36). Ideally we want to only generate the constraints that intersect at the optimal point. Suppose at iteration u we have generated the iterates $\mu_1, \ldots, \mu_u$. At each iterate μ_q, $q = 1, \ldots, u$ we generate the information:

$$
(2.37) \qquad z_{\mathrm{D}}^p = z_{\mathrm{D}}^p(\mu_q), \quad s_p = s_p(\mu_q) \quad p = 1, \ldots, |P_E(\mu_q)|,
$$

where $P_E(\mu_q)$ is an index set that contains at least one "optimal subgradient" corresponding to an optimal solution to (2.12)-(2.20) and then any "suboptimal subgradient" corresponding to any suboptimal solution to (2.12)-(2.20). This approach is equivalent to "multiple pricing" in column generation in which one may choose not only the non-basic variable with the most negative reduced cost but a set of non-basic variables with negative reduced costs [4]. Note that it is possible to control the number of subgradients generated $|P_E(\mu_q)|$ by adjusting E but we simply fix the maximum number of returned gradients to the dimension of μ.

To generate the next sampling point μ_{u+1} we maximize the cutting-plane model of our original problem (2.34):

$$
\begin{aligned}
(2.38) \qquad & \hat{z}_{\mathrm{LD}}^u(\mu) = \text{maximize}\, \theta \\
& \text{subject to} \\
& \theta \leqslant z_{\mathrm{D}}^p(\mu_q) + \langle s_p, \mu - \mu_q \rangle \quad \text{for } q = 1, \ldots, u \quad p = 1, \ldots, |P_E(\mu_q)|,
\end{aligned}
$$

but we add bounds on the variables, which is similar to the bounds used by Griffith and Stewart [16]:

$$
\begin{aligned}
(2.39) \qquad & \mu \leqslant \lambda_u + \Delta_u, \\
& \mu \geqslant \lambda_u - \Delta_u,
\end{aligned}
$$

where λ_u is the current solution at iteration u, which we refer to as the stability center. Instability refers to if the current iterate is closer (with respect to some norm) to the optimal solution than the next iterate. We refer to the parameter Δ_u as the trust-region since we assume the cutting-plane approximation of the dual function is good within this region. The idea of imposing the bounds of Griffith and Stewart [16] is in a nonlinear programming context used by Madsen [27] in an algorithm for minimizing a function that is expressed as the maximum of a finite number of explicit nonlinear functions. Madsen [27] also gives rules for updating Δ_u that we will fit to the case of linear functions we are considering. Independently,

Marsten et al. [29] also imposes the bounds of Griffith and Stewart [16] in the context of maximizing a concave function but do not assume that the function considered is explicitly available. However the box is kept constant, i.e. Δ_u is not updated. The authors conclude that the method may find fruitful application in Lagrangian relaxation and indeed our method can be viewed as an extension of the BOXSTEP method.

When we have computed μ_{u+1} we compute the decrease δ_u of our model (2.38) and (2.39):

$$\delta_u = \hat{z}^u_{\mathrm{LD}}(\mu_{u+1}) - z^1_D(\lambda_u),$$

where $z^1_{\mathrm{D}}(\lambda_u)$ is the current best dual solution value and $\hat{z}^u_{\mathrm{LD}}(\mu_{u+1})$ the LP-solution value of (2.38) and (2.39). Now, if δ_u is smaller than some user provided tolerance δ we have found an δ-optimal solution. Marsten et al. [29] shows that in the case of linear functions δ may be zero.

Next we solve the Lagrangian problem with respect to the sampling point μ_{u+1} and get the information:

$$z^p_{\mathrm{D}}(\mu_{u+1}), \quad s_p = s_p(\mu_{u+1}), \quad \text{for all } p \in P_E(\mu_{u+1}).$$

The trust-region parameter is adjusted automatically, depending on the ratio between the actual decrease of the dual function and the decrease predicted by the cutting-plane model (2.38) and (2.39):

$$\rho = \frac{z^1_{\mathrm{D}}(\mu_{u+1}) - z^1_{\mathrm{D}}(\lambda_u)}{\delta_u}.$$

If ρ is equal to 1 then we have just taken a step along one of the pieces of the dual function and no new information has been included. In this case we increase the size of the trust-region in order to quickly discover new constraints. If ρ is less than zero we have taken a step into a region where our current model is not a good approximation of the dual function and we decrease the trust-region in order to collect additional information in this region.

The algorithm is an ascent method since we use the condition $\rho > 0.01$ in order to move to the next sampling point μ_{u+1}, i.e. to take a serious step. If the ascent condition is not satisfied we stay at the current point, i.e. we take a null-step. The complete algorithm is as follows.

Algorithm 1 (Stabilized cutting-plane Algorithm) Choose an initial point λ_1, a stopping tolerance $\delta \geqslant 0$ and a trust-region size $\Delta_1 > 0$. Initialize the iteration-counter $u = 1$ and $\mu_1 = \lambda_1$; compute $z^p_{\mathrm{D}}(\mu_1)$ and s_p, for all $p \in P_E(\mu_1)$.

STEP 1 (Master problem). Solve the following relaxation of the dual problem:

$$\begin{aligned} (2.40) \qquad & \hat{z}^u_{\mathrm{LD}}(\mu) = \text{maximize } \theta \\ & \text{subject to} \\ & \theta \leqslant z^p_{\mathrm{D}}(\mu_q) + \langle s_p, \mu - \mu_q \rangle \quad \text{for } q = 1, \ldots, u \quad p = 1, \ldots, |P_E(\mu_q)|, \\ & \mu \leqslant \lambda_u + \Delta_u, \\ & \mu \geqslant \lambda_u - \Delta_u, \end{aligned}$$

to get a solution μ_{u+1} and compute:

$$\delta_u = \hat{z}^u_{\mathrm{LD}}(\mu_{u+1}) - z^1_{\mathrm{D}}(\lambda_u).$$

STEP 2 (Stopping criterion). If $\delta_u \leqslant \delta$ then stop.

STEP 3 (Local problem). Find a solution to the Lagrangian problem to get $z^p_{\mathrm{D}}(\mu_{u+1})$ and s_p, for all $p \in P_E(\mu_{u+1})$.

STEP 4 (Update). Compute the gain ratio $\rho = (z_D^1(\mu_{u+1}) - z_D^1(\lambda_u))/\delta_u$.
If $\rho = 1$ then set $\Delta_{u+1} = \Delta_u * 1.5$.
If $\rho < 0$ then set $\Delta_{u+1} = \Delta_u / 1.1$.
If $0 < \rho < 1$ then set $\Delta_{u+1} = \Delta_u$.
If $\rho > 0.01$ then set $\lambda_{u+1} = \mu_{u+1}$ (ascent-step). Otherwise set $\lambda_{u+1} = \lambda_u$ (null-step).
Set $u = u + 1$ and go to step 1. □

This type of algorithm is called a trust-region algorithm, introduced by Levenberg [26] and Marquardt [28] in connection with least squares calculations. For a recent presentation on ways to stabilize the cutting-plane algorithm we refer to Hiriart-Urruty and Lemaréchal [17, Chapter XV].

2.6 The Lagrangian branch-and-cut-and-price algorithm

In order to find integer solutions we embedded the stabilized cutting-plane algorithm in the open-source framework ABACUS [20]. ABACUS is a C++ class library for solving mixed-integer linear-programs by branch, cut, and price. Note that because we are using the dual Lagrangian based decomposition principle we perform cutting instead of pricing and vice versa compared to the primal Dantzig-Wolfe decomposition principle. ABACUS provides a general interface to linear programming solvers and we used the ILOG CPLEX solver (www.ilog.com). The overall solution procedure can be described as follows. First we solve the root which corresponds to solving the dual problem. If the dual optimum corresponds to an integer solution then we have found the optimal solution to the VRPTW. Else we start to generate strong valid inequalities. We generate subtour elimination constraints (SECS) and 2-path cuts using separation algorithms developed by Kohl [22]. However, the strong valid inequalities are added as columns in our linear program and the dual variables corresponding to the valid inequalities are also stabilized. We generate 2-path cuts only in the root node and SECS in all nodes of the branch-and-bound tree. If the generation of valid inequalities in the root node does not terminate with an integer solution branching is performed on the number of vehicles and on the x_{ijk} variables of the original formulation (2.1)-(2.10) [25]. We set the dual multipliers of child nodes to the optimal values from their parent node. In ABACUS each node in the enumeration tree also inherits the final constraint and variable system of the father node. This avoids tedious recomputations and is the reason why a cutting-plane method is superior to a simple subgradient based method. However we modify this system according to the branching rule on the arc variables since paths can be viewed as locally valid constraints in our method.

2.7 Computational results

The LBCP algorithm presented in this paper is implemented using standard C++ except the SPPTWCC algorithm developed by Larsen [25] and the separation algorithms for the SECS and 2-path cuts developed by Kohl [22], which are implemented in standard C. The computational experiments were performed on two different machines. The hardware and software configuration of the machines is given in Table 2.1. The LBCP algorithm was tested on the 56 Solomon [34] problems with 100 customers. This set of test problems was enlarged by only considering the first 25 and 50 customers of each original problem. This brings the total number of problems up to 168. In addition we have also tested the algorithm on a number of the problems created by Gehring and Homberger [15], who extended the Solomon problems to sizes of up to 1000 customers. In the Solomon and Homberger test problems the nodes $i = 0, \ldots, n$ are specified by integer coordinates (x_i', y_i') in the plane and the vehicle capacity by an integer q. For each node $i = 0, \ldots, n$ the following integer values are given: d_i, a_i', b_i', and st_i'. In order to fulfill the assumptions stated in this paper regarding the model parameters in (2.1)-(2.10) we perform the following transformations of the test problem data [22]. Step 1. Create a copy of node 0 and call it $n+1$. Step 2. For $i = 0, \ldots, n+1$ set

Machine	Sun Fire 15K	Dell Inspiron 7500
CPU	UltraSPARC III Cu 900MHz	Intel Pentium III 600MHz
RAM	384Gb	256Mb
Operating system	Solaris	Microsoft Windows XP
Compiler	g++ 2.95	Microsoft Visual C++ 6.0
Compiler options	-O	NDEBUG, Maximize Speed
ABACUS version	2.4	2.3
ILOG CPLEX version	8.1	7.5

Table 2.1: Hardware and software configuration for the computational experiments.

$x_i = 10x'_i$, $y_i = 10y'_i$, $a_i = 10a'_i$, $b_i = 10b'_i$, and $st_i = 10st'_i$. Step 3. $c_{ij} = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor$ and $t_{ij} = st_i + c_{ij}$ for $i, j = 0, \ldots, n+1$, $i \neq j$. Step 4. Add 1 to all c_{ij} for $i \neq 0$ in order to fulfill the triangle inequality. In steps 3 and 4 we have for notational convenience assumed that we have a fully connected digraph on the $n+2$ nodes. Clearly in the algorithms we only consider arcs $(i, j) \in A$. The solution value for the original problem is calculated as $(z_{\text{VRPTW}} - n)/10$, where z_{VRPTW} denotes the solution value for the transformed problem. Note that we also apply the preprocessing of the time windows described in Desrochers et al. [10].

2.7.1 Comparison of column generation and stabilized cutting-planes

First we wish to compare the numerical efficiency of the stabilized cutting-plane algorithm to an unstabilized column generation algorithm by solving respectively the Lagrangian dual problem (2.23) and the relaxed set-partitioning problem (2.25). This corresponds to solving the root node in the branch-and-bound tree. All computational experiments described in this section were performed on the Sun Fire 15K machine.

Let B denote the basis of the constraint matrix of (2.25) and c_{BS} the cost coefficients of the basis variables y_{BS}.

The column generation algorithm is as follows.

Algorithm 2 (Column generation Algorithm) The Dantzig-Wolfe master problem (2.25) is initialized with a feasible basis. Initialize the column counter $r = n$, the iteration counter $u = 1$ and compute the initial simplex multipliers $\pi = c_{\text{BS}} B^{-1}$.

STEP 1 (Subproblem). Return a solution of SPPTWCC (and any negative cost non-dominated path in the sink depot) with respect to the modified costs $\tilde{c}_{ij} = c_{ij} - \pi_i$ to obtain candidate columns y_p, with reduced costs $\bar{z}_p$, for $p \in P_u$.

STEP 2 (Stopping criterion). If $\bar{z}_p \geqslant -\delta$, for $p \in P_u$, then stop (all variables price out correctly). Otherwise adjoin the $|P_u|$ columns with negative reduced cost to the restricted master problem and set $r = r + |P_u|$.

STEP 3 (Master problem). Compute a solution to (2.25), i.e. determine a new basis and calculate the new simplex multipliers $\pi = c_{\text{BS}} B^{-1}$, set $u = u + 1$ and go to step 1. □

Assume we have initialized (2.25) with the paths $\{(0, i, n+1)\}$, $\forall i \in C$, and suppose $c_p = 10000$ for $p = 0, \ldots, n-1$, then $\pi_i = 10000$ for $i \in C$. Figure 2.1 illustrates the effect of the size of the multipliers on the computational difficulty of the SPPTWCC subproblems. In the Dantzig-Wolfe column generation algorithm the multipliers are large, compared to the optimal level, in the beginning of the solution process, while the multipliers in the cutting-plane algorithm are small. As would be expected the number of labels

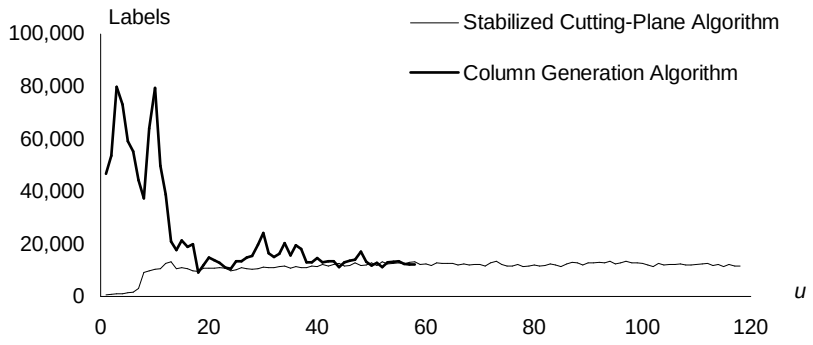


Figure 2.1: Development of the number of labels generated in the SPPTWCC subproblem against the iteration number when solving the dual problem (2.23) for a 100 customer Solomon problem (R104.100).

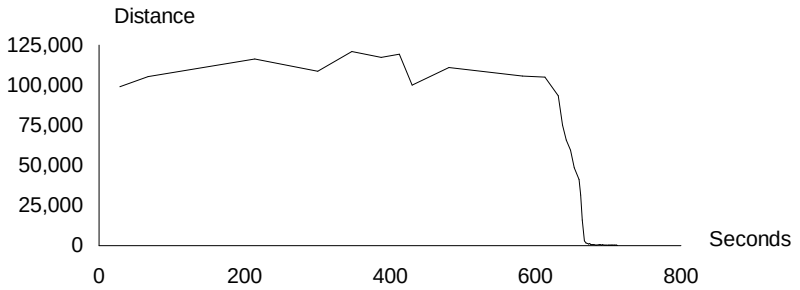
generated for solving the subproblems in the two algorithms are almost the same when the optimal level is reached.

Figure 2.2 illustrates the instability of the column generation algorithm compared to the stabilized cutting-plane algorithm. We have observed the same behavior as illustrated in figure 2.1 and 2.2 when solving several other problems. In fact, the total computational time for solving the dual problems for the R1, C1 and RC1 problems with 100 customers was decreased respectively by a factor 15, 46, and 2 by using the stabilized cutting-plane algorithm compared to the column generation algorithm. We present the results in Table 2.2-2.4, which show the following: Name of Solomon problem with suffix indicating the number of customers (Problem), value of lower bound (LB1), number of times the subproblem is solved (Iterations), and total CPU-time in seconds to reach the specified stopping tolerance (Seconds). The stabilized cutting-plane algorithm uses the initial information $\lambda_1 = \underline{0}$, $\delta = 10^{-7}$ and $\Delta_1 = 1$. The maximum number of constrained paths with negative reduced cost returned in each call of the SPPTWCC algorithm is fixed to the number of customers. For the column generation algorithm we used the stopping tolerance $\delta = 10^{-7}$ and the maximum number of constrained paths returned after each execution of the SPPTWCC algorithm was fixed to 100.

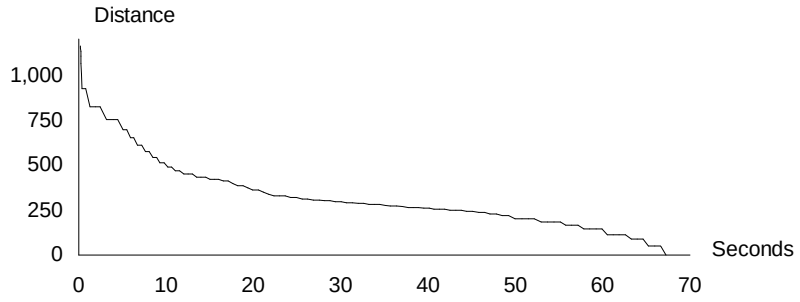
2.7.2 Solutions for the Solomon problems

Next we perform computational experiments with the LBCP algorithm on the Solomon problems. Our experiments were performed on the Sun Fire 15K machine. The reason for this is that we initially experienced extensive memory use in ABACUS 2.3 when the number of subproblems becomes large. ABACUS 2.4 provides a new interface to ILOG CPLEX 8 which solves the memory problem. However currently this version is available on Unix platforms only. Given the amount of memory available on the Sun Fire 15K machine memory is not a restriction in our experiments. However we restricted the maximal CPU time to 1 h. Making use of the stabilized cutting-plane algorithm in the branch-and-bound search we were able with the one hour restriction to solve 119 out of the 168 Solomon test problems. Table 2.5 gives an overview of the results compared to leading algorithms in the literature. We present a detailed overview of the results in Table 2.7-2.12 and describe the associated columns in Table 2.6.

Compared to our method the additional R1 and RC1 solutions in the literature are found using extensive parallel branching [5] or 3-cycle elimination [19]. The additional R2 and RC2 solutions are found using 3-, 4-, or 5-cycle elimination [19] or elementary shortest paths [2]. The C204.100 problem is solved by Irnich and Villeneuve [19] using 2-cycle elimination. However the computational time is more than



(a) Column Generation Algorithm



(b) Stabilized Cutting-Plane Algorithm

Figure 2.2: The Euclidian distance between the current dual variables and the optimum dual variables. Observe the different scales.

		Column Generation		Cutting-Plane	
Problem	LB1	Iterations	Seconds	Iterations	Seconds
R101.100	1631.150	30	1.36	85	1.08
R102.100	1466.600	48	42.08	102	4.70
R103.100	1206.312	64	183.82	143	20.05
R104.100	949.134	57	782.75	122	67.22
R105.100	1346.142	33	4.95	46	1.47
R106.100	1226.440	50	78.21	81	9.76
R107.100	1051.844	58	1538.29	108	31.38
R108.100	907.162	51	628.50	83	67.35
R109.100	1130.587	42	40.46	87	7.40
R110.100	1048.482	40	150.35	79	22.87
R111.100	1032.028	51	178.84	71	17.45
R112.100	919.192	41	872.47	59	45.56
Total		565	4502.08	1066	296.29

Table 2.2: Comparing the stabilized cutting-plane algorithm to a column generation algorithm on R1 problems with 100 customers.

		Column Generation		Cutting-Plane	
Problem	LB1	Iterations	Seconds	Iterations	Seconds
C101.100	827.300	83	8.53	53	0.84
C102.100	827.300	123	45.59	30	1.15
C103.100	826.300	115	172.05	55	5.58
C104.100	822.900	82	1195.66	73	18.50
C105.100	827.300	92	9.61	47	0.93
C106.100	827.300	85	7.77	34	0.81
C107.100	827.300	113	18.59	32	0.71
C108.100	827.300	92	16.09	47	1.44
C109.100	825.640	55	17.50	42	2.20
Total		840	1491.39	413	32.16

Table 2.3: Comparing the stabilized cutting-plane algorithm to a column generation algorithm on C1 problems with 100 customers.

		Column Generation		Cutting-Plane	
Problem	LB1	Iterations	Seconds	Iterations	Seconds
RC101.100	1584.094	31	1.92	50	1.13
RC102.100	1403.646	37	13.82	98	7.23
RC103.100	1218.495	44	50.46	72	19.76
RC104.100	1094.333	47	321.49	112	148.09
RC105.100	1471.160	38	6.94	106	6.82
RC106.100	1308.781	31	7.77	81	6.96
RC107.100	1170.689	35	42.67	58	21.23
RC108.100	1063.011	44	133.77	78	77.44
Total		307	578.84	655	288.66

Table 2.4: Comparing the stabilized cutting-plane algorithm to a column generation algorithm on RC1 problems with 100 customers.

Author	R1	C1	RC1	Type 1	R2	C2	RC2	Type 2	Total
Cook and Rich [5]	33	27	20	80	8	20	2	30	110
Irnich and Villeneuve [19]	29	27	20	76	21	24	14	59	135
Chabrier [2]					19	24	17	60	
This paper	28	27	18	73	17	23	6	46	119
Total solved	34	27	22	83	21	24	17	62	145
Total number of problems	36	27	24	87	33	24	24	81	168

Table 2.5: The number of Solomon problems solved.

Problem	Name of Solomon problem with suffix indicating number of customers.
LB1	Value of root node before any strong valid inequalities are generated. For some problems we were not able to solve the root node to optimality. This is indicated by a blank entry.
LB2	Value of root node after strong valid inequalities are generated. Only printed if LB2>LB1.
LB3	Value of global lower bound in the search tree when optimization was terminated. Only printed if LB3>max{LB1, LB2}. If the problem is solved max{LB1, LB2, LB3 } is equal to the optimal integer solution value.
Vehicles	Number of vehicles corresponding to UB.
B&B nodes	Number of selected nodes in the branch-and-bound tree.
Cuts	Number of generated strong valid inequalities in the root node and the search tree.
Iterations	Number of times the SPPTWCC algorithm is executed.
Seconds	Total CPU-time in seconds to solve problem to optimality (-*: time limit of 3600 CPU seconds exceeded).

Table 2.6: Description of columns in Table 2.7-2.12 and 2.17.

10 h. Compared to Irnich and Villeneuve [19] and Chabrier [2] the conclusion is that our algorithm is not competitive with respect to the R2 and RC2 problems, since the bound provided by the SPPTWCC with 2-cycle elimination is not tight enough. However compared to Cook and Rich [5] the acceleration strategy made it possible to solve more and larger problems in the R2 and RC2 sets.

Irnich and Villeneuve [19] performed all their computational experiments on a Pentium III 600MHz machine with 512Mb RAM. The Dell Inspiron 7500 machine is similar except that it only has 256Mb RAM. We should therefore be able to compare computational times for experiments performed on the Dell Inspiron 7500 with the results reported in Irnich and Villeneuve [19]. We tried to solve the 119 problems on the Dell Inspiron 7500 machine. However because of lack of memory we solved only 117 out of the 119 problems (R112.50 and R205.50 were not solved); 91 out of the 117 solved problems were solved faster than the minimum time reported in Irnich and Villeneuve [19] corresponding to a total decrease of 20437 seconds; 26 were solved with a longer computational time corresponding to a total increase of 940 seconds. Note that Irnich and Villeneuve [19] report solution times using 2-, 3-, and 4-cycle elimination. We compare our method with the *k*-cycle elimination method with the minimum solution time. In Table 2.13 we compare the total solution times per problem set for the LBCP algorithm with Irnich and Villeneuve [19]. In Table 2.15 and 2.16 we show the 10 problems for which we observed respectively the largest increase and decrease in the solution time by our method compared to the minimal solution time reported in Irnich and Villeneuve [19]. We describe the associated columns in Table 2.14.

2.7.3 Solutions for the Homberger problems

We have solved 9 problems from the Homberger test sets introduced in Gehring and Homberger [15], among them problems with 400 and 1000 customers. In 8 of the problems the customers are clustered (C-problems), while we succeeded in solving a 200 customer problem where the customers are randomly located. The results are presented in Table 2.17.

2.8 Conclusions

The algorithm has been tested on the Solomon VRPTW test problems and a range of extended Solomon problems created by Homberger. Using a stabilized cutting-plane algorithm in a branch-and-bound scheme gives a significant speed-up compared to an algorithm based on traditional column generation. We have solved two Homberger problem with 400 and 1000 customers respectively, which to date is the largest problems ever solved to optimality. The conclusion is therefore that it is an efficient acceleration strategy that performs significantly better than a traditional column generation based algorithm on a large number of test problems.

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
R101.25	617.100			8	1	0	30	0.1
R101.50	1043.367	1044.000		12	1	2	43	0.2
R101.100	1631.150	1634.000	1637.700	20	15	9	385	8.4
R102.25	546.333	547.100		7	1	3	32	0.1
R102.50	909.000			11	1	0	53	0.5
R102.100	1466.600			18	1	0	99	4.8
R103.25	454.600			5	1	0	39	0.2
R103.50	765.950	767.300	772.900	9	57	7	516	10.0
R103.100	1206.312	1206.376	1208.700	14	69	3	821	123.0
R104.25	416.900			4	1	0	32	0.2
R104.50	616.500	620.758	625.400	6	103	13	1129	303.9
R104.100	949.134	950.987	956.496	100	158	10	4057	-*
R105.25	530.500			6	1	0	24	0.1
R105.50	892.120	893.650	899.300	9	23	15	181	1.9
R105.100	1346.142	1348.632	1355.300	15	151	23	1573	102.9
R106.25	457.300	465.400		5	1	15	41	0.3
R106.50	791.367	793.000		8	1	7	57	0.9
R106.100	1226.440	1227.404	1234.600	13	1457	11	15173	2187.5
R107.25	422.925	423.800	424.300	4	3	3	41	0.3
R107.50	704.438	704.814	711.100	7	83	3	776	21.5
R107.100	1051.844	1052.714	1061.883	100	421	12	9568	-*
R108.25	396.139	396.720	397.300	4	3	2	61	0.5
R108.50	588.926	595.624	611.785	50	209	33	4854	-*
R108.100	907.162	910.603	915.265	100	234	17	4927	-*
R109.25	441.300			5	1	0	21	0.1
R109.50	775.096	775.890	786.800	8	247	7	1904	25.8
R109.100	1130.587	1133.164	1141.979	100	1011	34	26904	-*
R110.25	437.300	437.938	444.100	5	25	5	263	1.4
R110.50	692.577	694.150	697.000	7	5	4	83	2.2
R110.100	1048.482	1049.939	1058.607	100	491	8	9892	-*
R111.25	423.788	424.583	428.800	4	5	3	83	0.5
R111.50	691.812	692.635	707.200	7	461	14	4348	114.5
R111.100	1032.028		1041.850	100	528	6	10646	-*
R112.25	384.200	385.391	393.000	4	13	25	208	6.6
R112.50	607.219	612.374	630.200	6	5263	36	65983	3166.4
R112.100	919.192	922.398	930.128	100	171	41	4063	-*

Table 2.7: Solution overview for the R1 problems.

Problem	LB1	LB2	Vehicles	B&B nodes	Cuts	Iterations	Seconds
C101.25	191.300		3	1	0	24	0.1
C101.50	362.400		5	1	0	19	0.2
C101.100	827.300		10	1	0	50	0.8
C102.25	190.300		3	1	0	20	0.2
C102.50	361.400		5	1	0	38	0.5
C102.100	827.300		10	1	0	27	1.2
C103.25	190.300		3	1	0	26	0.2
C103.50	361.400		5	1	0	39	1.1
C103.100	826.300		10	1	0	52	5.7
C104.25	186.900		3	1	0	27	0.4
C104.50	357.250	358.000	5	1	3	67	7.3
C104.100	822.900		10	1	0	70	18.9
C105.25	191.300		3	1	0	26	0.1
C105.50	362.400		5	1	0	18	0.2
C105.100	827.300		10	1	0	44	0.9
C106.25	191.300		3	1	0	25	0.1
C106.50	362.400		5	1	0	28	0.2
C106.100	827.300		10	1	0	31	0.8
C107.25	191.300		3	1	0	24	0.1
C107.50	362.400		5	1	0	22	0.2
C107.100	827.300		10	1	0	29	0.7
C108.25	191.300		3	1	0	26	0.2
C108.50	362.400		5	1	0	34	0.3
C108.100	827.300		10	1	0	44	1.6
C109.25	191.300		3	1	0	25	0.2
C109.50	362.400		5	1	0	27	0.4
C109.100	825.640	827.300	10	1	3	104	6.1

Table 2.8: Solution overview for the C1 problems.

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
RC101.25	406.625	461.100		4	1	9	53	0.2
RC101.50	850.021	942.625	944.000	8	3	38	99	1.9
RC101.100	1584.094	1617.276	1619.800	15	27	68	340	32.9
RC102.25	351.800			3	1	0	22	0.2
RC102.50	719.902	813.037	822.500	7	683	12	6804	115.0
RC102.100	1403.646	1437.000	1450.327	100	1152	40	30698	-*
RC103.25	332.050		332.800	3	3	0	42	0.4
RC103.50	643.133	710.667	710.900	6	5	10	118	4.3
RC103.100	1218.495	1241.705	1250.706	100	423	44	12506	-*
RC104.25	305.825		306.600	3	7	0	75	0.7
RC104.50	543.750		545.800	5	17	5	247	18.4
RC104.100	1094.333	1112.354		100	3	32	360	-*
RC105.25	410.950		411.300	4	3	0	46	0.3
RC105.50	754.443	852.858	855.300	8	33	23	441	7.3
RC105.100	1471.160	1509.800	1513.700	15	41	34	686	65.6
RC106.25	342.829	343.200	345.500	3	13	1	135	0.6
RC106.50	664.433	714.788	723.200	6	37	12	532	11.9
RC106.100	1308.781	1332.510	1347.342	100	927	46	22511	-*
RC107.25	298.300			3	1	0	40	0.4
RC107.50	591.476	632.336	642.700	6	123	7	2095	193.6
RC107.100	1170.689	1178.484	1193.337	100	392	39	8850	-*
RC108.25	293.791	294.500		3	1	5	48	0.9
RC108.50	538.957	596.867	598.100	6	9	10	144	29.8
RC108.100	1063.011	1091.555	1096.177	100	95	52	2319	-*

Table 2.9: Solution overview for the RC1 problems.

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
R201.25	460.100		463.300	4	3	0	45	0.2
R201.50	788.425	791.900		6	1	1	67	0.8
R201.100	1136.248	1138.650	1143.200	8	183	2	3000	253.4
R202.25	406.350	408.350	410.500	4	5	1	87	0.8
R202.50	692.737	696.525	698.500	5	11	2	291	11.4
R202.100	1009.828	1009.859	1012.776	100	46	12	2143	-*
R203.25	379.882	381.625	391.400	3	37	3	426	5.4
R203.50	590.930	593.430	605.300	5	491	20	7906	923.4
R203.100	846.489	847.097	847.379	100	6	6	507	-*
R204.25	333.075	335.350	355.000	2	777	42	11052	190.6
R204.50	474.562	482.324	487.949	50	64	38	3874	-*
R204.100				100	1	0	378	-*
R205.25	381.283	388.425	393.000	3	15	4	205	1.1
R205.50	666.604	672.350	690.100	4	5255	34	98061	2558.7
R205.100	916.976	923.025	931.290	100	179	34	9884	-*
R206.25	363.132	365.908	374.400	3	85	12	946	14.2
R206.50	609.590	611.363	624.260	50	617	54	25518	-*
R206.100	835.326	840.751	844.619	100	16	12	1170	-*
R207.25	347.592	349.741	361.600	3	125	20	1568	30.2
R207.50	539.067	544.324	554.200	50	194	44	8821	-*
R207.100				100	1	13	588	-*
R208.25	318.105	318.911	328.200	1	75	24	1531	58.2
R208.50	462.412	471.563	472.779	50	14	30	1184	-*
R208.100				100	1	0	222	-*
R209.25	353.875	358.321	370.700	2	65	8	781	6.5
R209.50	582.926	588.413	600.600	4	525	49	9077	530.4
R209.100	819.847	823.734	829.175	100	45	28	2582	-*
R210.25	395.844	397.906	404.600	3	71	8	760	7.5
R210.50	624.421		638.524	50	1086	67	35729	-*
R210.100	849.477	855.852	860.328	100	21	17	1231	-*
R211.25	330.140	330.472	350.900	2	1513	84	23178	515.3
R211.50	507.950	512.567	523.607	50	235	58	7923	-*
R211.100	705.811	710.753		100	3	25	401	-*

Table 2.10: Solution overview for the R2 problems.

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
C201.25	214.700			2	1	0	48	0.2
C201.50	360.200			3	1	0	103	1.1
C201.100	589.100			3	1	0	40	2.0
C202.25	214.700			2	1	0	139	1.3
C202.50	360.200			3	1	0	574	20.3
C202.100	589.100			3	1	0	32	5.7
C203.25	214.700			2	1	0	102	5.7
C203.50	359.800			3	1	0	808	203.1
C203.100	588.700			3	1	0	77	73.9
C204.25	211.004	211.042	213.100	1	13	4	364	46.0
C204.50	350.100			2	1	0	471	402.2
C204.100				100	4	1	277	-*
C205.25	212.050	214.700		2	1	3	86	0.4
C205.50	357.350	359.000	359.800	3	3	1	400	7.2
C205.100	586.400			3	1	0	47	4.6
C206.25	197.700	214.700		2	1	6	112	0.9
C206.50	344.200	359.000	359.800	3	5	5	989	33.9
C206.100	585.400	586.000		3	1	2	64	14.9
C207.25	207.981	214.400	214.500	2	7	12	208	3.7
C207.50	356.269	359.600		3	1	13	236	22.3
C207.100	581.969	585.800		3	1	6	429	74.6
C208.25	193.280	214.500		2	1	10	165	1.8
C208.50	340.425	350.500		2	1	4	190	8.4
C208.100	581.767	585.800		3	1	4	159	57.0

Table 2.11: Solution overview for the C2 problems.

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
RC201.25	356.650	360.200		3	1	1	39	0.2
RC201.50	670.150	681.983	684.900	5	15	5	285	3.0
RC201.100	1240.398	1253.430	1261.800	9	679	18	16461	1205.9
RC202.25	290.408	313.389	338.000	3	665	30	12772	222.3
RC202.50	504.140	548.224	572.352	50	705	80	42784	-*
RC202.100	1004.398	1013.888	1028.253	100	126	30	6683	-*
RC203.25	214.475	260.811	295.866	25	984	111	49000	-*
RC203.50	409.246	480.052	486.953	50	23	42	2942	-*
RC203.100	815.276	831.971	834.207	100	7	38	714	-*
RC204.25	188.593	244.810	260.494	25	127	39	8907	-*
RC204.50				50	1	0	199	-*
RC204.100				100	1	0	196	-*
RC205.25	307.600	320.788	338.000	3	23	6	791	6.9
RC205.50	541.592	579.905	617.816	50	1093	59	84589	-*
RC205.100	1056.111	1070.859	1087.132	100	162	32	10709	-*
RC206.25	250.106	288.983	324.000	3	503	39	14526	195.5
RC206.50	441.336	532.192	551.372	50	375	73	35262	-*
RC206.100	952.406	982.887	995.911	100	126	40	7947	-*
RC207.25	217.961	263.894	292.133	3	2427	98	123256	-*
RC207.50	390.837	468.865	475.274	50	15	38	1693	-*
RC207.100	866.668	877.849	888.931	100	47	36	2988	-*
RC208.25	169.671	233.078	243.122	25	56	34	4095	-*
RC208.50				50	1	21	408	-*
RC208.100				100	1	13	389	-*

Table 2.12: Solution overview for the RC2 problems.

Author	R1	C1	RC1	Type 1	R2	C2	RC2	Type 2	Total
Irnich and Villeneuve [19]	3783.1	1011.8	2651.1	7446.0	5847.1	10146.9	3673.2	19667.2	27113.2
This paper	2550.0	48.2	397.1	2995.3	1981.0	1155.6	1484.5	4621.2	7616.4
Speed-up	1.5	21.0	6.7	2.5	3.0	8.8	2.5	4.3	3.6

Table 2.13: Total solution time in seconds for each problem set for the 117 problems solved by the LBCP algorithm on the Dell Inspiron 7500 vs. the corresponding minimum solution time reported by Irnich and Villeneuve [12].

Problem	Name of Solomon problem with suffix indicating number of customers.
LB1	Value of root node before any strong valid inequalities are generated.
LB2	Value of root node after strong valid inequalities are generated.
B&B nodes	Number of selected nodes in the branch-and-bound tree.
k	k-cycle elimination used in obtaining minimum the solution time.
Seconds	Total CPU-time in seconds to solve problem to optimality.

Table 2.14: Description of columns in Table 2.15 and 2.16.

	Irnich and Villeneuve [19]					LBCP Algorithm				
Problem	LB1	LB2	B&B nodes	k	Seconds	LB1	LB2	B&B nodes	Seconds	Increase
R104.50	616.500	620.758	34	2	212.6	616.500	620.758	117	406.9	194.3
R203.50	598.350	598.350	23	3	470.4	590.930	593.430	479	661.7	191.3
RC202.25	308.033	321.425	7	3	12.6	290.408	313.389	741	196.6	184.0
RC206.25	318.950	318.950	9	4	12.3	250.163	288.983	517	194.2	181.9
R209.50	599.813	599.813	7	4	255.4	582.877	588.413	531	421.1	165.7
R206.25	373.600	373.600	7	3	5.1	363.132	365.908	85	15.7	10.6
RC205.25	338.000	338.000	1	3	1.5	307.600	320.788	23	6.5	5.0
C203.25	214.700	214.700	1	2	3.7	214.700	214.700	1	4.9	1.2
RC106.25	345.500	345.500	1	3	0.4	342.829	343.200	13	1.2	0.8
R101.25	617.100	617.100	1	2	0.1	617.100	617.100	1	0.7	0.6

Table 2.15: Top 10 increases in computational time using the LBCP algorithm compared to Irnich and Villeneuve [12].

	Irnich and Villeneuve [19]					LBCP Algorithm				
Problem	LB1	LB2	B&B nodes	k	Seconds	LB1	LB2	B&B nodes	Seconds	Decrease
R201.100	1140.300	1140.300	35	4	3537.2	1136.222	1138.650	203	281.6	3255.6
RC201.100	1255.770	1256.260	39	4	3620.4	1240.398	1253.430	679	1082.7	2537.7
C208.100	581.767	585.800	2	2	2183.3	581.767	585.800	1	37.4	2145.9
C207.100	581.969	585.800	5	2	2068.8	581.969	585.800	1	52.1	2016.7
C203.100	588.700	588.700	1	2	1706.3	588.967	588.967	7	509.0	1197.3
RC105.100	1471.160	1509.800	25	2	899.0	1471.160	1509.800	41	56.4	842.6
C204.50	350.100	350.100	1	2	1159.4	350.100	350.100	1	332.8	826.6
C206.100	586.000	586.000	1	3	814.4	585.400	586.000	1	11.6	802.8
C202.100	589.100	589.100	1	2	585.6	589.100	589.100	1	6.5	579.1
R211.25	339.981	339.981	139	4	876.3	330.140	330.477	1127	335.3	541.0

Table 2.16: Top 10 decreases in computational time using the LBCP algorithm compared to Irnich and Villeneuve [12].

Problem	LB1	LB2	LB3	Vehicles	B&B nodes	Cuts	Iterations	Seconds
R1_2_1.200	4654.913	4661.033	4667.200	23	521	21	7980	1363.39
C1_2_1.200	2698.600			20	1	0	127	7.7
C1_2_2.200	2682.187		2694.300	20	75	6	2816	992.61
C1_2_5.200	2694.900			20	1	0	112	9.64
C1_2_6.200	2694.900			20	1	0	126	14.15
C1_2_7.200	2694.900			20	1	0	105	11.18
C1_2_8.200	2667.870	2668.118	2684.000	20	137	19	2409	601.15
C1_4_1.400	7138.767	7138.800		40	1	1	97	28.59
C110_1.1000	42444.400	42444.683	42444.800	100	5	3	858	1298.21

Table 2.17: Overview of the solved Homberger problems.

Acknowledgements The authors would like to thank Professor Kaj Madsen from the department of Informatics and Mathematical Modelling, Technical University of Denmark, for all the fruitful discussions regarding the development of Algorithm 1.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming Series A*, 90:475–506, 2001. [32]

[2] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. Technical report, ILOG, 2003. [33, 42, 44, 45]

[3] E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959. [33]

[4] V. Chvátal. *Linear Programming*. Freeman, New York, 1983. [38]

[5] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, Rice University, 1999. [33, 42, 44, 45]

[6] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 157–193. SIAM, Philadelphia, 2002. [32, 36]

[7] G. B. Dantzig and P. Wolfe. A decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960. [33]

[8] M. Desrochers. An algorithm for the shortest path problem with resource constraints. Technical report, GERAD, 1988. [37]

[9] M. Desrochers and F. Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26:191–211, 1988. [37]

[10] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992. [32, 36, 37, 41]

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994. [33, 37]

[12] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999. [33]

[13] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Technical report, Laboratoire d'Informatique d'Avignon, Université d'Avignon, 2004. [33]

[14] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981. [33]

[15] H. Gehring and J. Homberger. A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research*, 18:35–47, 2001. [33, 40, 45]

[16] R. E. Griffith and R. A. Stewart. A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, 7:379–392, 1961. [38, 39]

[17] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I-II*. Grundlehren der matematischen Wissenschaften 304-305. Springer-Verlag, Berlin Heidelberg, 1993. [33, 38, 40]

[18] D. J. Houck, J. C. Picard, M. Queyranne, and R. R. Vemuganti. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *OPSEARCH*, 17: 93–109, 1980. [37]

[19] S. Irnich and D. Villeneuve. The shortest path problem with k-cycle elimination ($k \geqslant 3$): Improving a branch-and-price algorithm for the VRPTW. Technical report, Lehr- und Forschungsgebiet Operations Research und Logistik Management, Rheinisch-Westfälische Technische Hochschule, 2003. [33, 42, 44, 45, 51, 52]

[20] M. Jünger and S. Thienel. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software: Practice and Experience*, 30:1325–1352, 2000. [40]

[21] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of SIAM*, 8:703–712, 1960. [33]

[22] N. Kohl. *Exact methods for Time Constrained Routing and Related Scheduling Problems*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1995. [40]

[23] N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean relaxation. *Operations Research*, 45:395–406, 1997. [32, 33, 37]

[24] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999. [33, 37]

[25] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1999. [37, 40]

[26] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly Applied Mathematics*, 2:164–168, 1944. [40]

[27] K. Madsen. An algorithm for minimax solution of overdetermined systems of non-linear equations. *Journal of the Institute of Mathematics and Its Applications*, 16:321–328, 1975. [38]

[28] D. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal of Applied Mathematics*, 11:431–441, 1963. [40]

[29] R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23:389–405, 1975. [39]

[30] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 49–78. SIAM, Philadelphia, 2002. [32]

[31] P. Neame, N. Boland, and D. Ralph. An outer approximate subdifferential method for piecewise affine optimization. *Mathematical Programming Series A*, 87:57–86, 2000. [37]

[32] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988. [37]

[33] M. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985. [32]

[34] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987. [33, 40]

Chapter 3

Path inequalities for the vehicle routing problem with time windows

Brian Kallehauge
Centre for Traffic and Transport, Technical University of Denmark

Natashia Boland
Department of Mathematics and Statistics, University of Melbourne

Oli B. G. Madsen
Centre for Traffic and Transport, Technical University of Denmark

Abstract

In this paper we introduce a new formulation of the vehicle routing problem with time windows (VRPTW) involving only binary variables associated with the arcs in the underlying digraph. The new formulation is based on the formulation of the asymmetric traveling salesman problem with time windows by Ascheuer et al. [3] and has the advantage of avoiding additional variables and linking constraints. In the new formulation of the VRPTW time windows are modeled using path inequalities. The path inequalities eliminate time and capacity infeasible paths. We present a new class of strengthened path inequalities based on the polyhedral results obtained by Mak [17] in the context of the asymmetric traveling salesman problem with replenishment arcs. We study the VRPTW polytope and determine the polytope dimension. We show that the lifted path inequalities are facet defining under certain assumptions. We also introduce precedence constraints in the context of the VRPTW. Computational experiments are performed with a branch-and-cut algorithm on the Solomon test problems with wide time windows. Based on results on 25-node problems the outcome is that the algorithm shows promising results compared to leading algorithms in the literature. In particular we report a solution to a previously unsolved 50-node Solomon test problem R208. The conclusion is therefore that the path formulation of Desrochers et al. [9] is no longer the unchallenged winning strategy for solving the VRPTW.

3.1 Introduction

This paper presents a polyhedral and computational study of a variant of the vehicle routing problem with time windows.

Throughout this paper, $D = (V,A)$ will be a directed graph, where $V = \{0,1,\ldots,n+1\}$ is the set of nodes, $|V| = n+2$, and A is the set of arcs. Node 0 and $n+1$ represent the start and destination depot, respectively. The remaining n nodes represent the set of customers $N = V \setminus \{0,n+1\}$. With every arc $(i,j) \in A$ a cost $c_{ij} \in \mathbb{Z}_+$ and time $t_{ij} \in \mathbb{Z}_+$ is associated. We assume that the triangle inequality on the costs and travel times is satisfied, i.e. $c_{ij} \leqslant c_{ih} + c_{hj}$ and $t_{ij} \leqslant t_{ih} + t_{hj}$ for all $(i,j) \in A$. With every node $j \in N$ we associate a demand $d_j \in \mathbb{Z}_+$, a release date $a_j \in \mathbb{Z}_+$, and a due date $b_j \in \mathbb{Z}_+$, where $a_j \geqslant t_{0j}$ and $b_j \geqslant a_j$ for all $j \in N$, and $b_j \geqslant a_i + t_{ij}$ for all $(i,j) \in A$. The release date a_i and the due date b_i are the earliest, respectively the latest possible starting time for servicing node i. For the nodes 0 and $n+1$ we assume that $d_0 = d_{n+1} = 0$, $a_0 = a_{n+1} = 0$, and $b_0 = b_{n+1} = +\infty$. With the graph D we associate a vehicle capacity $q \in \mathbb{Z}_+$, where $q \geqslant d_i$ for all $i \in N$, and $q \geqslant d_i + d_j$ for all $(i,j) \in A$. As usual, for each $W \subset V$ let $\delta^-(W) = \{(i,j) \in A \mid i \in V \setminus W, j \in W\}$, $\delta^+(W) = \{(i,j) \in A \mid i \in W, j \in V \setminus W\}$, and $A(W) = \{(i,j) \in A \mid i,j \in W\}$. We assume that $A \cap \{(0,n+1)\} = \emptyset$, $|\delta^+(0)| = |\delta^-(n+1)| = n$ and $\delta^-(0) = \delta^+(n+1) = \emptyset$. Let $m = |A(N)|$ so $|A| = 2n+m$.

A k-route consists of a partition $\{N_i \mid i = 1,\ldots,k\}$ of the set of customer nodes N into k subsets, and an associated sequence, or route, of each subset $r_i = (v_1,\ldots,v_{|N_i|})$ specifying the order of service of the customers. A route r_i represents a vehicle leaving the start depot 0, servicing the set of customers N_i in the order defined by r_i and entering the destination depot $n+1$. Note that a partition of the customer nodes may correspond to several k-routes. Let $\mathbb{R}_+^N$ be the set of nonnegative real vectors whose components are indexed by N. With each k-route we associate a vector $s \in \mathbb{R}_+^N$, defined as follows: $s_{v_1} = a_{v_1}$ and $s_{v_j} = \max\{s_{v_{j-1}} + t_{v_{j-1}v_j}, a_{v_j}\}$ for $j = 2,\ldots,|N_i|$ and $i = 1,\ldots,k$. We say that a k-route is time and capacity feasible if $s_{v_j} \leqslant b_{v_j}$ for $j = 1,\ldots,|N_i|$ and $\sum_{j \in N_i} d_j \leqslant q$ for $i = 1,\ldots,k$. We say that the partition size k, or vehicle fleet size, is feasible, if a corresponding feasible k-route exists and we denote by $k(N)$ the minimum feasible fleet size. We denote by $K = \{k(N),\ldots,|N|\}$ the set of feasible fleet sizes. For each $k \in K$ we denote by $\mathscr{R}_k$ the set of feasible k-routes using exactly k vehicles. The set of feasible k-routes for all $k \in K$ is denoted $\mathscr{R}_K = \{\cup \mathscr{R}_k \mid k = k(N),\ldots,|N|\}$. Let $R_k \in \mathscr{R}_K$ denote any feasible k-route and let $c(R_k)$ be the corresponding cost defined as $c(R_k) = (\sum c(r_i) \mid i = 1,\ldots,k)$ where $c(r_i) = c_{0v_1} + (\sum c_{v_{j-1}v_j} \mid j = 2,\ldots,|N_i|) + c_{v_{|N_i|}n+1}$. The vehicle routing problem with time windows is

(VRPTW) $\quad \min\{c(R_k) \mid R_k \in \mathscr{R}_K\}$.

The VRPTW reduces to the capacitated vehicle routing problem (CVRP) if $a_i = 0$ and $b_i = +\infty$ for every $i \in N$. Therefore the VRPTW is NP-hard. Indeed, it is strongly NP-complete to find a feasible solution for the VRPTW with a fixed number of vehicles [19].

Bard et al. [6] present an integer linear programming formulation of a different variant of the VRPTW where the objective is to minimize the number of vehicles. In this formulation binary variables are associated with arcs in the underlying digraph. Integer node variables are introduced to model the time and capacity restrictions following the approach of Miller et al. [18]. In the context of the asymmetric traveling salesman problem with time windows (ATSPTW) Ascheuer et al. [3, 4] modeled time window restrictions using "infeasible path elimination" constraints and they noted that similar constraints can also be used to model any other kind of path infeasibility. We follow this suggestion and present a new formulation of the VRPTW involving binary arc variables only. In our formulation time and capacity restrictions are modeled using infeasible path elimination constraints (IPECs), which we denote path inequalities. We present a class of strengthened path inequalities based on the polyhedral results obtained by Mak [17] in the context of the asymmetric traveling salesman problem with replenishment arcs (RATSP). Our formulation is exponential in size since there are exponentially many path inequalities. We present the first polyhedral results on the VRPTW polytope by proving that the new class of strengthened path inequalities are facet-defining for the VRPTW polytope under certain conditions. We solve the problem by a branch-and-cut algorithm based on the new path inequalities and other classes of inequalities adopted from the asymmetric traveling salesman problem (ATSP) and the precedence constrained ATSP [5].

The reasons for this line of research were the following. First, the polyhedral approach to the related

ATSPTW indicates that most instances in the range of up to 50-70 nodes can be solved to optimality via branch-and-cut codes in a few minutes [3]. Second, we believe that the polyhedral approach may outperform the dynamic programming approach of the path formulation when the time windows are wide. Third, the results of Bard et al. [6] indicate that in their case instances with 50 nodes can be routinely solved using a branch-and-cut code. Finally, no polyhedral results on the VRPTW polytope were known when this investigation started and further research in this area could show what the relative advantages of the different solution methods are.

This paper reports some computational results on VRPTW test problems developed by Solomon [20]. We focus on test problems having a long planning horizon and large vehicle capacity making it possible for each vehicle to visit many customers on a route.

The paper is organized as follows. In Section 3.2 we present a new binary integer programming (BIB) formulation of the VRPTW and determine the dimension of the associated polytope. In Section 3.3 we prove that a new class of strengthened path inequalities define a proper face of the VRPTW polytope and that the inequalities are facet defining under certain conditions. In Section 3.4 we describe how classes of valid inequalities for the precedence constrained ATSP polytope can be transferred to the VRPTW. Section 3.6 reviews preprocessing routines for the VRPTW based on time windows and their implications. In Section 3.7 we describe the test problems providing the data for our computational study and the computational platform for performing the experiments. Section 3.8 is dedicated to the implementational details of the branch-and-cut algorithm. In Section 3.9 we report extensive computational results following the approach of Ascheuer et al. [4]. Finally, we present our conclusions in Section 3.10.

3.2 A BIP formulation of the VRPTW

A path in the graph D is a sequence of nodes $P = (v_1, \ldots, v_p)$ such that $(v_i, v_{i+1}) \in A$ for all $i = 1, \ldots, p-1$. Let $V(P)$ and $A(P)$ denote respectively the set of nodes and arcs of the path. The path is always open and simple, i.e. $|A(P)| = p-1$ and $v_i \neq v_j$ for $i \neq j$. We say that a path P is infeasible if it does not occur as a subpath in any feasible route, i.e. if either $\sum_{i=1}^{p} d_{v_i} > q$ or $s_{v_i} > b_{v_i}$, for some $i \in \{1, \ldots, p\}$. An infeasible path P is said to be minimal infeasible if the truncated subpaths defined by $A(P) \setminus \{(v_1, v_2)\}$ and $A(P) \setminus \{(v_{p-1}, v_p)\}$ are feasible. We denote by $\mathscr{P}_D$ the set of all minimal infeasible paths in D. A necessary and sufficient condition for a k-route to be feasible is that no route that defines it contains any minimal infeasible path.

Let $\mathbb{R}_+^A$ be the set of nonnegative real vectors whose components are indexed by A; the incidence vector of a k-route in $\mathscr{R}_K$ is $x \in \mathbb{R}_+^A$, defined as follows: $x_{ij} = 1$ if $(i, j) \in A$ is used in the k-route and $x_{ij} = 0$ otherwise. For notational convenience we do not distinguish between a k-route and its incidence vector. For any $Q \subseteq A$, we write $x(Q)$ for $\sum_{(i,j)\in Q} x_{ij}$. With this notation, the set of feasible k-routes $\mathscr{R}_K$ for the VRPTW is the set of those k-routes for which the incidence vector $x \in \mathbb{B}^A$ satisfy the degree equations

$$x(\delta^+(i)) = 1, \quad \forall i \in N, \tag{3.1}$$

$$x(\delta^-(i)) = 1, \quad \forall i \in N, \tag{3.2}$$

the subtour inequalities

$$x(A(W)) \leqslant |W| - 1, \quad \forall \emptyset \neq W \subseteq N, \tag{3.3}$$

and the path inequalities

$$x(A(P)) \leqslant |A(P)| - 1, \quad \forall P \in \mathscr{P}_D. \tag{3.4}$$

We denote by $\mathscr{P}_{\text{VRPTW}} \subset \mathbb{R}_+^A$ the VRPTW polytope, i.e. the convex hull of the incidence vectors of all the k-routes in $\mathscr{R}_K$, $\mathscr{P}_{\text{VRPTW}} = \text{conv}\{\mathscr{R}_K\}$. As for the dimension of the VRPTW polytope, we have $\dim(\mathscr{P}_{\text{VRPTW}}) = m$.

Proposition 3.2.1 $\dim(\mathscr{P}_{\text{VRPTW}}) = m$.

Proof: Clearly $\dim(\mathscr{P}_{\text{VRPTW}}) \leqslant |A| - 2n = m$, since $\mathscr{P}_{\text{VRPTW}} \subset \mathbb{R}^A$ and the system (3.1) and (3.2) of degree equations has rank $2n$.

Next we show that there are $m+1$ affinely independent k-routes in $\mathscr{P}_{\text{VRPTW}}$. First consider the k-route $R_n \in \mathscr{R}_K$ consisting of n routes, one for each customer

$$R_n = \{(0, i, n+1) \mid i \in N\},$$

and then consider the following m k-routes denoted by $R_{n-1}(i,j)$, each of them containing an arc (i,j) in $A(N)$ which is not used by any other of the $m-1$ k-routes or by the k-route R_n, thus proving the affine independence.

$$R_{n-1}(i,j) = \{(0,i,j,n+1)\} \cup \{(0,i',n+1) \mid i' \in N \setminus \{i,j\}\}, \quad \forall (i,j) \in A(N).$$

This gives a total of $m+1$ affinely independent k-routes and so $\dim(\mathscr{P}_{\text{VRPTW}}) \geqslant m$. Therefore, from $\dim(\mathscr{P}_{\text{VRPTW}}) \leqslant m$ and $\dim(\mathscr{P}_{\text{VRPTW}}) \geqslant m$ it follows that $\dim(\mathscr{P}_{\text{VRPTW}}) = m$. □

3.3 Lifted path inequalities

The path inequalities (3.4) can be very weak. Mak [17] proposes ways to strengthen path inequalities for the ATSP with Replenishment Arcs (RATSP) and introduces a class of valid inequalities she called S_1 constraints. In what follows we transfer Mak's S_1 inequalities to the VRPTW adopting the definitions used therein.

Definition 3.3.1 Suppose we have a minimal infeasible path $P = (v_1, \ldots, v_p) \in \mathscr{P}_D$. Then we define the following arc sets.

1. $\Delta^+(P) = \bigcup_{h=1}^{p-1} \Delta^+_{v_h}(P)$ is the set of non-path arcs starting at nodes in P, where $\Delta^+_{v_h}(P) = \{(v_h, j) \in A \mid j \neq v_{h+1}\}$, for $h = 1, \ldots, p-1$.

2. $\widetilde{\Delta}^+(P) = \bigcup_{h=1}^{p-1} \widetilde{\Delta}^+_{v_h}(P)$ is the set of time and capacity feasible forward and escaping non-path arcs w.r.t. the minimal infeasible path P, where $\widetilde{\Delta}^+_{v_h}(P) = \{(v_h, j) \in A(N) \mid j \neq v_1, \ldots, v_{h+1},\ \sum_{l=1}^{h} d_{v_l} + d_j \leqslant q \text{ and } s_{v_h} + t_{v_h,j} \leqslant b_j\} \cup \{(v_h, n+1) \in A\}$, for $h = 1, \ldots, p-1$.

3. $\overline{\Delta}^+(P) = \bigcup_{h=1}^{p-1} \overline{\Delta}^+_{v_h}(P)$ is the set of time and capacity infeasible forward and escaping non-path arcs w.r.t. the minimal infeasible path P, where $\overline{\Delta}^+_{v_h}(P) = \{(v_h, j) \in A(N) \mid j \neq v_1, \ldots, v_{h+1},\ \sum_{l=1}^{h} d_{v_l} + d_j > q \text{ or } s_{v_h} + t_{v_h,j} > b_j\}$, for $h = 1, \ldots, p-1$. Since all arcs $(i,j) \in A$ are time and capacity feasible we have $\Delta^+_{v_1}(P) = \emptyset$.

4. $\Delta^B(P) = \bigcup_{h=1}^{p-1} \Delta^B_{v_h}(P)$ is the set of backward arcs w.r.t. the minimal infeasible path P, where $\Delta^B_{v_h}(P) = \{(v_h, j) \in A(N) \mid j \in \{v_1, \ldots, v_{h-1}\}\}$, for $h = 2, \ldots, p-1$ and $\Delta^B_{v_1}(P) = \emptyset$.

5. $\delta^+(v_h) = \widetilde{\Delta}^+_{v_h}(P) \cup \overline{\Delta}^+_{v_h}(P) \cup \Delta^B_{v_h}(P) \cup \{(v_h, v_{h+1})\}$ is the set of all arcs leaving node v_h, for $h = 1, \ldots, p-1$.

The outdegree equations (3.1) give us a means to alter the form of the path inequalities (3.4) (the "inside" form). By subtracting the outdegree equations $x(\delta^+(i)) = 1$ for the nodes $i \in \{v_1, \ldots, v_{p-1}\}$ of an infeasible path in (3.4) and dividing the inequality by -1 we obtain the "outside" form

$$x(\Delta^+(P)) \geqslant 1 \quad \forall P \in \mathscr{P}_D. \tag{3.5}$$

From definition 3.3.1, we get

$$\Delta^+(P) = \widetilde{\Delta}^+(P) \cup \overline{\Delta}^+(P) \cup \Delta^B(P),$$

and therefore (3.5) is equivalent to

$$x(\widetilde{\Delta}^+(P)) + x(\overline{\Delta}^+(P)) + x(\Delta^B(P)) \geqslant 1 \quad \forall P \in \mathscr{P}_D.$$

Inequalities (3.5) can be lifted to give

$$x(\widetilde{\Delta}^+(P)) \geqslant 1 \quad \forall P \in \mathscr{P}_D, \tag{3.6}$$

and we now prove that this is a valid class of inequalities for the VRPTW.

Proposition 3.3.1 For $P \in \mathscr{P}_D$, the path inequality

$$x(\widetilde{\Delta}^+(P)) \geqslant 1 \tag{3.7}$$

is valid for the VRPTW polytope.

Proof: Assume to the contrary that there exists an $x \in \mathscr{R}_K$ such that (3.7) does not hold, i.e. $x(\widetilde{\Delta}^+(P)) = 0$ and so $x(\widetilde{\Delta}^+_{v_h}(P)) = 0$, for $h = 1, \ldots, p-1$. We now prove by induction that if (3.7) does not hold, then $x_{v_h, v_{h+1}} = 1$, for $h = 1, \ldots, p-1$, contradicting our assumption that $x \in \mathscr{R}_K$.

Initial step. When $h = 1$ the outdegree constraint for node v_1 gives us $x(\delta^+(v_1)) = x(\widetilde{\Delta}^+_{v_1}(P)) + x(\overline{\Delta}^+_{v_1}(P)) + x(\Delta^B_{v_1}(P)) + x_{v_1 v_2} = 1$. By assumption we have $x(\widetilde{\Delta}^+_{v_1}(P)) = 0$ and by definition we have $\Delta^B_{v_1}(P) = \emptyset$, hence $x(\Delta^B_{v_1}(P)) = 0$. Recall that $\overline{\Delta}^+_{v_1}(P) = \emptyset$ and therefore $x(\overline{\Delta}^+_{v_1}(P)) = 0$. So for $h = 1$ $x_{v_h v_{h+1}} = 1$ is true.

Inductive step. Assume there is a t, for $2 \leqslant t \leqslant p-1$, such that $x_{v_l v_{l+1}} = 1$ is true for $l = 1, \ldots, t-1$. We now prove that $x_{v_l v_{l+1}} = 1$ is true for $l = t$. When $l = t$ the outdegree constraint for node v_t gives us $x(\delta^+(v_t)) = x(\widetilde{\Delta}^+_{v_t}(P)) + x(\overline{\Delta}^+_{v_t}(P)) + x(\Delta^B_{v_t}(P)) + x_{v_t v_{t+1}} = 1$. Again, by assumption we have $x(\widetilde{\Delta}^+_{v_t}(P)) = 0$ and by the inductive assumption $x_{v_l v_{l+1}} = 1$, for $l = 1, \ldots, t-1$, and therefore we must have $\Delta^+_{v_t}(P) = \emptyset$, otherwise the solution would induce a minimal infeasible path. Now, assume there is an l, for $1 \leqslant l \leqslant t-1$, such that $x_{v_t v_l} = 1$. Then by the inductive assumption the solution would induce a subtour and therefore we must have $x(\Delta^B_{v_t}(P)) = 0$. Now, from the outdegree constraint we have $x_{v_t v_{t+1}} = 1$. This completes the inductive step and hence the proposition is proved.

□

The inside form of the inequality (3.7) is

$$x(P) + x(\overline{\Delta}^+(P)) + x(\Delta^B(P)) \leqslant |A(P)| - 1. \tag{3.8}$$

Definition 3.3.2 (Mak [17]) Given a time and capacity constrained digraph D and a minimal infeasible path $P \in \mathscr{P}_D$ we denote by $S_1^{D,P}$ the constraint (3.8).

Ascheuer et al. [3] propose ways to strengthen path inequalities for the ATSPTW and introduce a class of lifted path inequalities they call tournament constraints. To begin we define additional arc sets.

Definition 3.3.3 Suppose we have a minimal infeasible path $P = (v_1, \ldots, v_p) \in \mathscr{P}_D$. Then we define the following arc sets.

1. $\Delta^E(P) = \bigcup_{h=1}^{p-1} \Delta^E_{v_h}(P)$ is the set of escaping non-path arcs w.r.t. the minimal infeasible path P, where $\Delta^E_{v_h}(P) = \{(v_h, j) \in A(N) \mid j \notin \{v_1, \ldots, v_p\}\} \cup \{(v_h, n+1) \in A\}$, for $h = 1, \ldots, p-1$
2. $\Delta^F(P) = \bigcup_{h=1}^{p-1} \Delta^F_{v_h}(P)$ is the set of non-path forward arcs w.r.t. the minimal infeasible path P, where $\Delta^F_{v_h}(P) = \{(v_h, j) \in A(N) \mid j = v_{h+2}, \ldots, v_p\}$, for $h = 1, \ldots, p-2$ and $\Delta^F_{v_{p-1}}(P) = \emptyset$.
3. $\delta^+(v_h) = \Delta^E_{v_h}(P) \cup \Delta^F_{v_h}(P) \cup \Delta^B_{v_h}(P) \cup \{(v_h, v_{h+1})\}$ is the set of all arcs leaving node v_h, for $h = 1, \ldots, p-1$.

We note that the relationship between the arc sets of definition 3.3.1 and 3.3.3 is given by

$$\widetilde{\Delta}^+(P) \cup \overline{\Delta}^+(P) = \Delta^E(P) \cup \Delta^F(P). \tag{3.9}$$

Given a $P \in \mathscr{P}_D$, Ascheuer et al. [3] denote by $[P] = \{(v_i, v_j) \in A \mid 1 \leqslant i \leqslant j \leqslant p\}$ the transitive closure of $P = (v_1, \ldots, v_p)$. Ascheuer et al. [3] show that for every $P \in \mathscr{P}_D$ the tournament constraint

$$x([P]) \leqslant |A(P)| - 1 \tag{3.10}$$

is valid for the ATSPTW polytope. Using definition 3.3.3 we can write (3.10) as

$$x(P) + x(\Delta^F(P)) \leqslant |A(P)| - 1. \tag{3.11}$$

Observe that (3.11) expands to

$$\sum_{h=1}^{p-1} \left(x_{v_h v_{h+1}} + x(\Delta^F_{v_h}(P)) \right) \leqslant |A(P)| - 1. \tag{3.12}$$

By the outdegree equation, this is equivalent to

$$\sum_{h=1}^{p-1} \left(1 - x(\Delta^E_{v_h}(P)) - x(\Delta^B_{v_h}(P)) \right) \leqslant |A(P)| - 1, \tag{3.13}$$

and so to

$$\sum_{h=1}^{p-1} \left(x(\Delta^E_{v_h}(P)) + x(\Delta^B_{v_h}(P)) \right) \geqslant 1, \tag{3.14}$$

and also

$$x(\Delta^E(P)) + x(\Delta^B(P)) \geqslant 1. \tag{3.15}$$

We now prove that (3.15) is a valid constraint for the VRPTW.

Proposition 3.3.2 For $P \in \mathscr{P}_D$, the path inequality

$$x(\Delta^E(P)) + x(\Delta^B(P)) \geqslant 1 \tag{3.16}$$

is valid for the VRPTW polytope.

Proof: Assume to the contrary that there exists an $x \in \mathscr{R}_K$ such that (3.16) does not hold, i.e. $x(\Delta^E(P)) + x(\Delta^B(P)) = 0$ and so $x(\Delta^E_{v_h}(P)) + x(\Delta^B_{v_h}(P)) = 0$, for $h = 1,\ldots,p-1$. We now prove by induction that if (3.16) does not hold, then $x_{v_{h-1},v_h} = 1$, for $h = p,\ldots,2$, contradicting our assumption that $x \in \mathscr{R}_K$.

Initial step. When $h = p$ the outdegree constraint for node v_{p-1} gives us $x(\delta^+(v_{p-1})) = x(\Delta^E_{v_{p-1}}(P)) + x(\Delta^F_{v_{p-1}}(P)) + x(\Delta^B_{v_{p-1}}(P)) + x_{v_{p-1},v_p} = 1$. By assumption we have $x(\Delta^E_{v_{p-1}}(P)) + x(\Delta^B_{v_{p-1}}(P)) = 0$ and by definition we have $\Delta^F_{v_{p-1}}(P) = \emptyset$, hence $x(\Delta^F_{v_{p-1}}(P)) = 0$. So for $h = p$ $x_{v_{h-1}v_h} = 1$ is true.

Inductive step. Assume there is a t, for $2 \leqslant t \leqslant p-1$, such that $x_{v_{l-1}v_l} = 1$ is true for $l = p,\ldots,t+1$. We now prove that $x_{v_{l-1}v_l} = 1$ is true for $l = t$. When $l = t$ the outdegree constraint for node v_t gives us $x(\delta^+(v_t)) = x(\Delta^E_{v_t}(P)) + x(\Delta^F_{v_t}(P)) + x(\Delta^B_{v_t}(P)) + x_{v_{t-1},v_t} = 1$. Again, by assumption we have $x(\Delta^E_{v_t}(P)) + x(\Delta^B_{v_t}(P)) = 0$ and by the inductive assumption $x_{v_{l-1}v_l} = 1$, for $l = p,\ldots,t+1$, and the indegree constraints $x(\delta^-(i)) = 1$, for $i = p,\ldots,t+1$ we must have $x(\Delta^F_{v_t}(P)) = 0$. Now, from the outdegree constraint we have $x_{v_{t-1}v_t} = 1$. This completes the inductive step and hence the proposition is proved.

3.3.1 Facet proof

Definition 3.3.4 Given any minimal infeasible path $P \in \mathscr{P}_D$, we define the face of $\mathscr{P}_{\text{VRPTW}}$ induced by $S_1^{D,P}$ to be $F_1^{D,P} = \{x \in \mathscr{P}_{\text{VRPTW}} \mid x(A(P)) + x(\Delta^+(P)) + x(\Delta^B(P)) = |A(P)| - 1\}$.

Lemma 1 For any time and capacity constrained digraph $D = (V,A)$ with $N = \{1,\ldots,n\}$ where $|N| \geqslant 5$, and any minimal infeasible path $P = (v_1,\ldots,v_p) \in \mathscr{P}_D$ with $3 \leqslant p \leqslant |N| - 2$, $F_1^{D,P}$ is a proper face of $\mathscr{P}_{\text{VRPTW}}$.

Proof: To show that $F_1^{D,P}$ is a proper face of $\mathscr{P}_{\text{VRPTW}}$, we need to show that $\emptyset \neq F_1^{D,P} \neq \mathscr{P}_{\text{VRPTW}}$. We first show that $\emptyset \neq F_1^{D,P}$ by showing that there is at least one solution in $\mathscr{R}_K$ that satisfies constraint $S_1^{D,P}$ at equality. Without loss of generality, we assume that $P = \{1,\ldots,p\}$. Consider a feasible solution $x^1 \in \mathscr{R}_K$ defined by $x^1_{ij} = 1$ if $(i,j) \in \{(k,k+1) \mid k = 1,\ldots,p-2\} \cup \{(0,j) \mid j \geqslant p\} \cup \{(j,n+1) \mid j \geqslant p\}$ and $x^1_{ij} = 0$ otherwise. Since this is clearly a feasible k-route using exactly $|A(P)| - 1$ arcs, constraint $S_1^{D,P}$ is satisfied at equality. Next we show that $F_1^{D,P} \neq \mathscr{P}_{\text{VRPTW}}$ by showing that there is at least one solution in $\mathscr{R}_K$ that does not satisfy constraint $S_1^{D,P}$ at equality. Consider another feasible solution $x^2 \in \mathscr{R}_K$ where $x^2_{ij} = 1$ if $(i,j) \in \{(0,j) \mid j = 1,\ldots,n\} \cup \{(j,n+1) \mid j = 1,\ldots,n\}$ and $x^2_{ij} = 0$ otherwise. Clearly, as none of the arcs in $A(P) \cup \overline{\Delta}^+(P) \cup \Delta^B(P)$ are used in the k-route, constraint $S_1^{D,P}$ is not satisfied at equality. □

Given a path $P \in \mathscr{P}_D$ let $\Omega(P) = \cup_{h=1}^{p-1} \Omega_{v_h}(P)$, where $\Omega_{v_h}(P) = \{k \in V \mid (j,k) \in \overline{\Delta}^+_{v_h}(P)\}$. Let ω_{v_h} denote any arbitrary node in the set $\Omega_{v_h}(P)$.

Theorem 3.3.1 For any time and capacity constrained digraph $D = (V,A)$ with $N = \{1,\ldots,p+2\}$ and any minimal infeasible path $P = (v_1,\ldots,v_p) \in \mathscr{P}_D$, constraint $S_1^{D,P}$ defines a facet for $\mathscr{P}_{\text{VRPTW}}$ if the following conditions hold:

1. all arcs in A are time and capacity feasible, i.e. $b_j \geqslant a_i + t_{ij}$ and $q \geqslant d_i + d_j$ for all $(i,j) \in A$,
2. if (i,v_h) exists in $A(N)$ for $i \in N \setminus V(P)$ and $v_h \in V(P) \setminus \{v_1,v_p\}$, then there must exist an arc $(v_h,\omega_{v_h}) \in A(N)$ and (i,v_h,ω_{v_h}) must be feasible,
3. if (v_1,v_h) exists in $A(N)$ for any $h \in \{3,\ldots,p-1\}$, then there must exist $(v_h,\omega_{v_h}) \in A(N)$ and $(v_{h-1},\omega_{v_{h-1}}) \in A(N)$ where ω_{v_l} is distinct for distinct l and (v_1,v_h,ω_{v_h}) and $(v_2,\ldots,v_{h-1},\omega_{v_{h-1}})$ must be feasible.

4. if (v_1, v_p) or (v_p, v_1) exists in $A(N)$, then there must exist $(v_{p-1}, \omega_{v_{p-1}}) \in A(N)$ and $(v_2, \ldots, v_{p-1}, \omega_{v_{p-1}})$ must be feasible.

5. if (v_h, v_1) exists in $A(N)$ for any $h \in \{3, \ldots, p-1\}$, then there must exist $(v_{h-1}, \omega_{v_{h-1}}) \in A(N)$ and $(v_2, \ldots, v_{h-1}, \omega_{v_{h-1}})$ must be feasible.

6. if (v_p, v_h) exists in $A(N)$ for any $h \in \{1, \ldots, p-1\}$, then there must exist $(v_h, \omega_{v_h}) \in A(N)$ and $(v_{p-1}, \omega_{v_{p-1}}) \in A(N)$ where ω_{v_l} is distinct for distinct l and (v_p, v_h, ω_{v_h}) and $(v_{h+1}, \ldots, v_{p-1}, \omega_{v_{p-1}})$ must be feasible.

7. if (v_h, v_p) exists in $A(N)$ for any $h \in \{2, \ldots, p-2\}$, then there must exist $(v_{p-1}, \omega_{v_{p-1}}) \in A(N)$ and $(v_{h+1}, \ldots, v_{p-1}, \omega_{v_{p-1}})$ must be feasible.

8. if (v_i, v_j) exists in $A(N)$, for $i, j \in \{2, \ldots, p-1\}$, $i < j-1$, then there must exist $(v_j, \omega_{v_j}) \in A(N)$ and $(v_{j-1}, \omega_{v_{j-1}}) \in A(N)$ where ω_{v_l} is distinct for distinct l and $(v_1, \ldots, v_i, v_j, \omega_{v_j})$ and $(v_{i+1}, \ldots, v_{j-1}, \omega_{v_{j-1}})$ must be feasible.

9. if (v_i, v_j) exists in $A(N)$, for $i, j \in \{2, \ldots, p-1\}$, $i > j$, then there must exist $(v_j, \omega_{v_j}) \in A(N)$ and $(v_{i-1}, \omega_{v_{i-1}}) \in A(N)$ where ω_{v_l} is distinct for distinct l and (v_i, v_j, ω_{v_j}) and $(v_{j+1}, \ldots, v_{i-1}, \omega_{v_{i-1}})$ must be feasible.

Proof: We now show that under the conditions of the lemma, $F_1^{D,P}$ has dimension $\dim(\mathscr{P}_{\text{VRPTW}}) - 1$. From proposition 1 we know that $F_1^{D,P}$ has dimension at most $\dim(\mathscr{P}_{\text{VRPTW}}) - 1$. We therefore have to show that the dimension of $F_1^{D,P}$ is at least $\dim(\mathscr{P}_{\text{VRPTW}}) - 1$. We show this directly by constructing $\dim(\mathscr{P}_{\text{VRPTW}})$ affinely independent feasible k-routes that satisfy constraint $S_1^{D,P}$ at equality. First we split $A(N)$ into the following disjoint sets:

$$
\begin{aligned}
A_1 &= A(P), \\
A_2 &= (V(P) \setminus \{v_1, v_p\}, V \setminus V(P)), \\
A_3 &= (V \setminus V(P), V(P) \setminus \{v_1, v_p\}), \\
A_4 &= (\{v_1, v_p\}, V \setminus V(P)), \\
A_5 &= (V \setminus V(P), \{v_1, v_p\}), \\
A_6 &= (\{v_1\}, V(P) \setminus \{v_1\}) \setminus A(P), \\
A_7 &= (V(P) \setminus \{v_1\}, \{v_1\}), \\
A_8 &= (\{v_p\}, V(P) \setminus \{v_1, v_p\}), \\
A_9 &= (V(P) \setminus \{v_1, v_p\}, \{v_p\}) \setminus A(P), \\
A_{10} &= (V(P) \setminus \{v_1, v_p\}, V(P) \setminus \{v_1, v_p\}) \setminus A(P), \text{ with } i < j \text{ for } (v_i, v_j) \in A_{10}, \\
A_{11} &= (V(P) \setminus \{v_1, v_p\}, V(P) \setminus \{v_1, v_p\}), \text{ with } i > j \text{ for } (v_i, v_j) \in A_{11}, \\
A_{12} &= (V \setminus V(P), V \setminus V(P)).
\end{aligned}
$$

Then for every A_l, $l = 1, \ldots, 12$, we construct a k-route for all $(i, j) \in A_l$ that satisfy constraint $S_1^{D,P}$ at equality. We also make sure that the arc $(i, j) \in A_l$ for which we construct a k-route is not contained in any of the other previously introduced k-routes, thereby ensuring the affine independence. Note that the k-routes constructed for A_1 are special because we actually ensure the affine independence by the absence of the arc $(i, j) \in A_1$ in the corresponding k-route.

1. $\{(v_1, \ldots, v_h)\} \cup \{(v_{h+1}, \ldots, v_p)\} \cup \{(v_k) \mid k \in N \setminus V(P)\}$, $\forall (v_h, v_{h+1}) \in A_1$.

2. (a) $\{(v_1, \ldots, v_h, j)\} \cup \{(v_{h+1}, \ldots, v_p)\} \cup \{(v_k) \mid k \in N \setminus \{V(P) \cup \{j\}\}\}$ if $(v_h, j) \notin \bar{\Delta}^+_{v_h}(P)$,

(b) $\{(v_1,\ldots,v_{h-1})\}\cup\{(v_h,j)\}\cup\{(v_{h+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{j\}\}\}$ otherwise, $\forall(v_h,j)\in A_2$.

3. $\{(i,v_h,\omega_{v_h})\}\cup\{(v_1,\ldots,v_{h-1})\}\cup\{(v_{h+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{i,\omega_{v_h}\}\}\}$, $\forall(i,v_h)\in A_3$.

4. (a) $\{(v_1,j)\}\cup\{(v_2,\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{j\}\}\}$ if $h=1$,
   (b) $\{(v_p,j)\}\cup\{(v_1,\ldots,v_{p-1})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{j\}\}\}$ otherwise, $\forall(v_h,j)\in A_4$.

5. (a) $\{(i,v_1)\}\cup\{(v_2,\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{i\}\}\}$ if $h=1$,
   (b) $\{(i,v_p)\}\cup\{(v_1,\ldots,v_{p-1})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{i\}\}\}$ otherwise, $\forall(i,v_h)\in A_5$.

6. (a) $\{(v_1,v_h,\omega_{v_h})\}\cup\{(v_2,\ldots,v_{h-1},\omega_{v_{h-1}}\}\cup\{(v_{h+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_h},\omega_{v_{h-1}}\}\}\}$ if $h\neq p$,
   (b) $\{(v_1,v_p)\}\cup\{(v_2,\ldots,v_{p-1},\omega_{v_{p-1}})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_{p-1}}\}\}\}$ otherwise, $\forall(v_1,v_h)\in A_6$.

7. (a) $\{(v_h,v_1)\}\cup\{(v_2,\ldots,v_{h-1},\omega_{v_{h-1}})\}\cup\{(v_{h+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_{h-1}}\}\}\}$ if $h\neq p$,
   (b) $\{(v_p,v_1)\}\cup\{(v_2,\ldots,v_{p-1},\omega_{v_{p-1}})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_{p-1}}\}\}\}$ otherwise, $\forall(v_h,v_1)\in A_7$.

8. $\{(v_p,v_h,\omega_{v_h})\}\cup\{(v_1,\ldots,v_{h-1})\}\cup\{(v_{h+1},\ldots,v_{p-1},\omega_{v_{p-1}})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_h},\omega_{v_{p-1}}\}\}\}$, $\forall(v_p,v_h)\in A_8$.

9. (a) $\{(v_h,v_p)\}\cup\{(v_1,\ldots,v_{h-1})\}\cup\{(v_{h+1},\ldots,v_{p-1},\omega_{v_{p-1}})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_{p-1}}\}\}\}$ if $(v_h,v_p)\in\overline{\Delta}^+_{v_h}(P)$,
   (b) $\{(v_1,\ldots,v_h,v_p)\}\cup\{(v_{h+1},\ldots,v_{p-1},\omega_{v_{p-1}})\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_{p-1}}\}\}\}$ otherwise, $\forall(v_h,v_p)\in A_9$.

10. $\{(v_1,\ldots,v_i,v_j,\omega_{v_j})\}\cup\{(v_{i+1},\ldots,v_{j-1},\omega_{v_{j-1}})\}\cup\{(v_{j+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_j},\omega_{v_{j-1}}\}\}\}$, $\forall(v_i,v_j)\in A_{10}$.

11. $\{(v_i,v_j,\omega_{v_j})\}\cup\{(v_1,\ldots,v_{j-1})\}\cup\{(v_{j+1},\ldots,v_{i-1},\omega_{v_{i-1}})\}\cup\{(v_{i+1},\ldots,v_p)\}\cup\{(v_k)\mid k\in N\setminus\{V(P)\cup\{\omega_{v_j},\omega_{v_{i-1}}\}\}\}$, $\forall(v_i,v_j)\in A_{11}$.

12. $\{(i,j)\}\cup\{(v_1,\ldots,v_{p-1})\}\cup\{(v_k)\mid k\in N\setminus\{i,j,v_1,\ldots,v_{p-1}\}\}$, $\forall(i,j)\in A_{12}$.

□

3.4 Precedence constraints

Time windows induce precedences among the nodes [4], that is, whenever a set of customers $N_l\in N$ are served on the same route $l\in\{1,\ldots,k\}$ and $b_i<a_j+t_{ji}$ for $i,j\in N_l$, we can conclude that i has to precede j in any feasible route. Let $i\prec j$ denote the fact that i has to precede j in any feasible route and let $P=(V,R)$ denote an additional precedence digraph defined on the node set $V=V$ of D. An arc $(i,j)\in R$ represents a precedence relationship $i\prec j$. Note that by definition $\delta^+(0),\delta^-(n+1)\in R$ and $R\cap\{(0,n+1)\}=\emptyset$. In the context of the ATSPTW which Ascheuer et al. [4] considers, the precedence digraph is acyclic and transitively closed. However neither is true for the VRPTW, because in our case the nodes $i\in N$ are not required to all be visited on the same route as in the ATSPTW, e.g. if a 2-cycle exists $(i-j-i)$ it means that the node pair (i,j) cannot belong to the same route and we refer to it as an incompatible pair. This term was introduced by Bard et al. [6]. Also, if $i\prec j$ and $j\prec k$ it does not mean

that $i \prec k$ because j may be served by a different vehicle than i, which means it may be possible for k to precede i on the same route. Now, let

$$\pi(v) = \{i \in V \mid (i,v) \in R\}, \tag{3.17}$$

$$\sigma(v) = \{j \in V \mid (v,j) \in R\}, \tag{3.18}$$

represent the set of the predecessors and successors of a node $v \in V$, respectively.

Balas et al. [5] derived families of valid inequalities for the precedence constrained asymmetric traveling salesman (PCATS) polytope that can be seen as strengthenings of the subtour inequalities (3.3). These inequalities can also be written in the equivalent cut form

$$x(W, \overline{W}) \geqslant 1 \quad \forall \emptyset \neq W \subseteq N \tag{3.19}$$

Now we present two strengthenings of (3.19) in the VRPTW context. The results we present follow easily from Balas et al. [5], but these strengthenings have not been considered previously in our context:

Proposition 3.4.1 For $W \subseteq N$ and any given $j \in W$, $\overline{W} = V \setminus W$, the weak predecessor-inequality (weak π-inequality)

$$x(W \setminus \pi(j), \overline{W} \setminus \pi(j)) \geqslant 1 \tag{3.20}$$

is valid for the VRPTW polytope.

Proof: Let r be any feasible route and let $j \in r$. Let $\hat{w}$ be the last node of W served by r, and note that $\hat{w} \in W \setminus \pi(j)$. The successor of $\hat{w}$ in r cannot be in $\pi(j)$. □

Balas et al. [5] presented a second class of inequalities, in which the successors of W play a role analogous to that of the predecessors of W in (3.20). The authors describe how the role of predecessor and successor can be switched by basically replacing the digraph D with the transposed $D^T = (V, A^T)$, where $A^T = \{(i,j) \in V \times V \mid (j,i) \in A\}$. This immediately gives us the following result.

Proposition 3.4.2 For $W \subseteq N$, and any given $j \in W$, $\overline{W} = V \setminus W$, the weak successor-inequality (weak σ-inequality)

$$x(\overline{W} \setminus \sigma(j), W \setminus \sigma(j)) \geqslant 1 \tag{3.21}$$

is valid for the VRPTW polytope.

Note that the general π- and $\sigma-$inequalities are not generally valid for the VRPTW, since they are derived from the fact that in the PCATSP context all nodes have to be served by the same vehicle. Balas et al. [5] also presented a predecessor-successor inequality or $\pi - \sigma$-inequality and a precedence cycle breaking inequality or pcb-inequality but it is easy to show that these families of inequalities are not generally valid for the VRPTW polytope. These inequalities are also derived from the fact that in the PCATSP context all nodes have to be served by the same vehicle. However, we can still make use of these families of inequalities when we are considering the VRPTW with exactly one vehicle:

(1-VRPTW) $\min\{c(R_1) \mid R_1 \in \mathscr{R}_1\}$.

The 1-VRPTW is an important subproblem in the solution of the VRPTW. The 1-VRPTW is identical with the ATSPTW if the total demand is not greater than the vehicle capacity, i.e. $\sum_{i \in N} d_i \leqslant q$. The strengthened precedence inequalities can be used in the branching strategy where nodes with the number of vehicles set to one is bounded using these inequalities, $x(N) = 1$. The inequalities can also be used

in the separation of the 2-path inequalities proposed by Kohl et al. [15], in which one has to determine whether it is feasible to serve a subset of customer nodes using exactly one vehicle. It is also possible to use these inequalities in the case constraint branching is performed on a subset of customers, $S \subset N$, dividing the problem into (i) $x(S) = 1$ and (ii) $x(S) \geqslant 2$. In subproblem (i) the strengthened inequalities are valid with respect to $S \subset N$, and the separation problem can be solved for this subset.

3.5 ATSP inequalities

In this section we describe the classes of inequalities from the ATSP that we use in our implementation of the branch-and-cut algorithm:

- Odd Closed Alternating Trail (CAT) inequalities [10]
  Two distinct arcs (i,j) and (u,v) are called incompatible if $i = u$, or $j = v$, or $i = v$ and $j = u$; compatible otherwise. Let $(i,j) \nleftrightarrow (u,v)$ denote the fact that (i,j) and (u,v) are incompatible and $(i,j) \leftrightarrow (u,v)$ that they are compatible. An odd CAT is a sequence

  $$T = \{a_1, \ldots, a_t\} \in A(N), \quad t \geqslant 3 \text{ and odd}, \tag{3.22}$$

  of t distinct arcs, such that for $k = 1, \ldots, t$

  $$\begin{aligned} &a_0 = a_t \\ &a_{t+1} = a_1 \\ &a_k \nleftrightarrow a_{k+1} \\ &a_k \nleftrightarrow a_{k-1} \\ &a_k \leftrightarrow a_i, \quad i = 1, \ldots, t,\ i \neq k+1, \text{ and } i \neq k-1. \end{aligned}$$

  We denote by $\mathscr{T}_D$ the set of all CATs in the directed graph D. For $T \in \mathscr{T}_D$ let

  $$s(N) = \{i \in N \mid |\delta^+(i) \cap T| = 2\} \tag{3.23}$$

  and

  $$t(N) = \{i \in N \mid |\delta^-(i) \cap T| = 2\}, \tag{3.24}$$

  denote the set of source and sink nodes of N, respectively. Moreover, let

  $$Q = \{(i,j) \in A(N) \setminus T \mid i \in s(N),\ j \in t(N)\}. \tag{3.25}$$

  For any $T \in \mathscr{T}_D$ the odd CAT inequality is

  $$x(T \cup Q) \leqslant \frac{|T| - 1}{2}. \tag{3.26}$$

- D_k^+-inequalities [10]

  $$x_{i_1 i_k} + \sum_{h=2}^{k} x_{i_h i_{h-1}} + 2 \sum_{h=2}^{k-1} x_{i_1 i_h} + \sum_{h=3}^{k-1} x(\{i_2, \ldots, i_{h-1}\}, i_h) \leqslant k - 1, \tag{3.27}$$

  where $(i_1, \ldots, i_k)$ is any sequence of $k \in \{3, \ldots, n-1\}$ distinct nodes.

- D_k^--inequalities [10]

  $$x_{i_k i_1} + \sum_{h=2}^{k} x_{i_{h-1} i_h} + 2 \sum_{h=2}^{k-1} x_{i_h i_1} + \sum_{h=3}^{k-1} x(i_h, \{i_2, \ldots, i_{h-1}\}) \leqslant k - 1, \tag{3.28}$$

  where $(i_1, \ldots, i_k)$ is any sequence of $k \in \{3, \ldots, n-1\}$ distinct nodes.

3.6 Preprocessing

For the VRPTW, preprocessing includes three main steps: tightening of the time windows, deducing from the time windows the precedence relationships among customer nodes, and eliminating arcs.

3.6.1 Tightening of the time windows

Kontoravdis and Bard [16] presented criteria (3.29) and (3.30) for removing the time windows of the depot nodes:

$$a_k = \max\{a_k, a_0 + t_{0k}\} \quad \forall k \in N, \tag{3.29}$$

$$b_k = \min\{b_k, b_{n+1} - t_{kn+1}\} \quad \forall k \in N. \tag{3.30}$$

Desrochers et al. [9] presented criteria (3.31)-(3.32) for increasing the release time of the time windows and criteria (3.33)-(3.34) for decreasing the due time of the time windows:

$$a_k = \max\{a_k, \min_{(i,k)\in A}\{a_i + t_{ik}\}\} \quad \forall k \in N \text{ s.t. } \delta^-(k) \neq \emptyset, \tag{3.31}$$

$$a_k = \max\{a_k, \min\{b_k, \min_{(k,j)\in A}\{a_j - t_{kj}\}\}\} \quad \forall k \in N \text{ s.t. } \delta^+(k) \neq \emptyset, \tag{3.32}$$

$$b_k = \min\{b_k, \max\{a_k, \max_{(i,k)\in A}\{b_i + t_{ik}\}\}\} \quad \forall k \in N \text{ s.t. } \delta^-(k) \neq \emptyset, \tag{3.33}$$

$$b_k = \min\{b_k, \max_{(k,j)\in A}\{b_j - t_{kj}\}\} \quad \forall k \in N \text{ s.t. } \delta^+(k) \neq \emptyset. \tag{3.34}$$

In a fully connected directed graph it is obvious that criterion (3.29) is equivalent to (3.31), i.e. $\{(0,k)\} \in \text{Argmin}_{(i,k)\in A}\{a_i + t_{ik}\}$, and criterion (3.30) is equivalent to (3.34), i.e. $\{(k,n+1)\} \in \text{Argmax}_{(k,j)\in A}\{b_j - t_{kj}\}$. However, if we do not have a graph with all possible arcs with head in 0 or tail in $n+1$ we need criterion (3.31) and (3.34). This is for example the case if we branch on the arc variables, i.e. we apply criterion (3.31) on the subproblem having $x_{0j} = 0$, $j \in N$, and (3.34) is used when $x_{in+1} = 0$, $i \in N$.

3.6.2 Precedence relationships and elimination of arcs

In Section 3.4 we described how the time windows implied certain precedence relationships. The construction of precedence relationships in the VRPTW follows the description in Ascheuer et al. [4] except that the precedence digraph in our case is neither acyclic nor transitively closed. The time windows of the VRPTW and the construction of precedence relationships moves our problem from a fully connected graph to a graph where certain arcs are eliminated. By construction, if (i, j) is in the set R of the precedence digraph, the arc (j,i) cannot be contained in any feasible solution to the VRPTW.

3.7 Test problems and computational platform

The test problems that we use in this paper were developed by Solomon [20] and are divided into two classes, these are denoted by the number 1 and 2. For each of the classes he generated three subclasses, these are denoted by the letter R, C, and RC. We present an overview of the test problems in Table 3.1. The

number of problems #P in class 1 and 2 is respectively 29 and 27; all problems contain $n = 100$ customer nodes denoted by $1,\ldots,n$ and a single depot node denoted by 0. We enlarged the set of test problems by only considering the first 25 and 50 customer nodes of each original problem. The nodes are specified by integer coordinates (x'_i, y'_i) in the Euclidean $[0,100]^2$ plane and the vehicle capacity by an integer q. All test problems have an upper bound on the number of vehicles of 25. For each node $i = 0,\ldots,n$ a demand d_i, start time a'_i, due time b'_i, and service time st'_i is specified. For the R, C, and RC subclasses the service times are respectively 10, 90, and 10 for all $i \in N$, 0 otherwise. All node parameters are integer values. The depot start time a'_i is 0 and all customer due times b'_i, $i \in N$, are less than the depot due time b_0. In order to fulfill the assumptions stated in Section 3.1 regarding the model parameters of the VRPTW we perform the following transformations of the test problem data: Step 1. Create a copy of node 0 and call it $n+1$, set $b_0 = 0$. Step 2. For $i = 0,\ldots,n+1$ set $x_i = 10\,x'_i$, $y_i = 10\,y'_i$, $a_i = 10\,a'_i$, $b_i = 10\,b'_i$, and $st_i = 10\,st'_i$. Step 3. Set $c_{ij} = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor$ and $t_{ij} = st_i + c_{ij}$ for $i,\ j = 0,\ldots,n+1$, $i \neq j$. Step 4. Add 1 to all c_{ij} for $i \neq 0$ in order to fulfill the triangle inequality; the service time is positive hence the triangle inequality on the travel times is also satisfied. Step 5. Apply the preprocessing steps of Section 3.6. The solution value for the original problem is calculated as $(c(\hat{R}_k) - n)/10$, where $c(\hat{R}_k)$ denotes the solution value for the transformed problem.

Subclasses R, C, and RC consists of problems where the distribution of nodes in the plane is respectively random, clustered, and semi-clustered. Problem class 1 consists of problems with a relatively small vehicle capacity q compared to the total customer demand; in problem class 2 the vehicle capacity is relatively large. Now we want to assess whether the length of the planning horizon $[0, b_{n+1}]$ is a constraining factor. As noted in Section 3.1 this can be seen as another constraint associated with the vehicles. We consider the complete graph $D_C = (A_C, V)$ on the $n+2$ nodes in V. We compute the Hamilton path in D_C starting in 0 and ending in $n+1$. Note that concerning subclasses R and RC the nodes are identically distributed in class 1 and 2, while this is not the case for subclass C. This means that in relation to subclasses R and RC the Hamilton path solution is identical for class 1 and 2; this is not true for subclass C. In Table 3.1 we present the depot due time b_{n+1} and the duration of the Hamilton path, i.e. the time the vehicle visits the destination depot $n+1$. Problem class 1 consists of test problems with a relatively short planning horizon compared to the duration of the Hamilton path while class 2 consists of problems with a relatively long planning horizon. In class 2 the 25- and 50-customer test problems in subclasses R and RC are not constrained by the vehicle capacity nor the length of the planning horizon, whereas in subclass C this is only true for the 25-customer problems. In class 1 all test problems are constrained by vehicle capacity and the length of the planning horizon.

Next we turn to the individual time constraints associated with the customers. Solomon designed two different methods for assigning time windows to customers. The first method was designed for the random generation of time windows and used in the subclasses R and RC. The second method was designed for assigning time windows in a structured way to the clustered problems in subclass C. In method 2 Solomon used a 3-opt method on each cluster to create routes and then generated time windows by choosing the center as the arrival time at each customer. The time windows generated in method 2 therefore allows for cluster-by-cluster solutions to the C-problems. In terms of the number of customers that received a time window smaller than the depot time window, Solomon created problems where 25, 50, 75, and 100% of the customers received such a time window. However, in the measurements we made the test problems R110, R111, R112, RC105, RC107, RC108, and R210 had a time window density of 86-99%. In Table 3.2 we present a grouping of the test problems based on their time windows density. Note that we have included the test problems in the range 86-99% in the 100% category.

The computational experiments were conducted on two different machines. The hardware and software configuration of the machines is given in Table 3.3. The branch-and-cut algorithm presented in this paper is implemented using the ILOG CPLEX Concert Technology for C++ and the ILOG CPLEX 9.1 Mixed Integer Optimizer [11]. In Section 3.8 we will describe implementational details of the branch-and-cut algorithm.

Subclass	#P	q	$\sum_{1\le i\le n} d_i$			b_{n+1}	Hamilton path duration		
			$n=25$	$n=50$	$n=100$		$n=25$	$n=50$	$n=100$
Problem class 1									
R	12	200	332	721	1458	230	562.3	959.7	1635.9
C	9	200	460	860	1810	1236	2381.8	4739.5	9501.1
RC	8	200	540	970	1724	240	475.6	869.6	1640.7
Total	29								
Problem class 2									
R	11	1000	332	721	1458	1000	562.3	959.7	1635.9
C	8	700	460	860	1810	3390	2445.5	4825.2	9542.3
RC	8	1000	540	970	1724	960	475.6	869.6	1640.7
Total	27								

Table 3.1: Solomon's test problems

	R	C	RC
Problem class 1			
100%	1, 5, 9, 10, 11, 12	1, 5, 6, 7, 8, 9	1, 5, 6, 7, 8
75%	2, 6	2	2
50%	3, 7	3	3
25%	4, 8	4	4
Problem class 2			
100%	1, 5, 9, 10, 11	1, 5, 6, 7, 8	1, 5, 6, 7, 8
75%	2, 6	2	2
50%	3, 7	3	3
25%	4, 8	4	4

Table 3.2: Time window density of Solomon's 100-customer test problems

Machine	Sun Fire 15K	Dell Inspiron 7500
CPU	UltraSPARC III Cu 900MHz	Intel Pentium III 600MHz
RAM	384Gb	256Mb
Operating system	Solaris	Microsoft Windows XP
Compiler	Sun Studio CC 9	Microsoft Visual C++ .NET 7.1
Compiler options	-fast -xarch=v8plusb	/O2 /MD
ILOG CPLEX version	9.1.0	9.1.0

Table 3.3: Hardware and software configuration for the computational experiments

3.8 The branch-and-cut algorithm

According to the ILOG CPLEX 9.1 User's Manual [11] the Mixed Integer Optimizer of CPLEX solves MIP problems using a general branch-and-cut algorithm. We therefore assume that the CPLEX algorithm follows the basic steps of a branch-and-cut algorithm described in e.g. Wolsey [21, Fig. 9.5]. In this section we describe the parts of the branch-and-cut algorithm which are specific for our approach to solving the VRPTW:

- Formulation of the initial binary integer program
- Separation routines
- Branching
- Enumeration strategy

The separation routines, branching, and enumeration strategy is implemented as three CPLEX callback functions, which are executed respectively in the CUT, BRANCHING, and NODE step of the branch-and-cut algorithm described in Wolsey [21, Fig. 9.5]. We have turned off the presolve routines and the general valid inequalities of CPLEX, i.e. clique cuts, Gomory fractional cuts etc. We also set CPLEX to emphasize optimality over feasibility. For all other parameters in CPLEX we use the default value. Note that our approach for solving the VRPTW could have been implemented in any branch-and-cut framework available, e.g. the open-source framework ABACUS [13]. The result of our implementational work is a basic Dantzig-Fulkerson-Johnson branch-and-cut (DFJBC) algorithm [2] for the VRPTW.

3.8.1 Formulation of the initial binary integer program

We generate all the variables $x_{ij} \in A$, the integer constraints $x \in \mathbb{B}^A$, and the degree constraints (3.1) and (3.2). Finally, we add a lower bound on the number of vehicles $x(N) \geqslant \lceil \sum_{i \in N} d_i / q \rceil$. A common approach to solving large-scale integer linear programs by branch-and-cut is to combine cutting-planes with the dual concept of column generation [8]. In CPLEX 9.1 it is not possible to combine cutting-planes and column generation, however, this is possible in ABACUS and other open-source frameworks.

3.8.2 Separation routines

In this section we describe the separation procedures for the class of inequalities used in our branch-and-cut algorithm.

- **Subtour, π-, σ-, (π,σ)-inequalities:** For the subtour and π-inequalities we use the algorithm proposed by Balas et al. [5] that simultaneously solves the separation problem for both the subtour inequalities and the (weak) π-inequalities. The maxflow step of the algorithm is implemented using the ILOG CPLEX 9.1 Network Optimizer, which solves a minimum cost flow problem using a network simplex method. The maxflow problem is therefore formulated as a minimum cost flow problem [1]. The overall complexity of the algorithm is $O(n^4)$. For the separation of σ- and (π,σ)-inequalities we also use the algorithms proposed by Balas et al. [5] with complexities of respectively $\min\{n, |R|\} \cdot O(n^3)$ and $O(|R| \cdot n^3)$.
- **Lifted path inequalities:** For the tournament constraints we use the enumeration procedure proposed in Ascheuer et al. [4]. We adopt this procedure for the separation of S_1-inequalities. We detect paths where $x^*(\widetilde{\Delta}^+(P))$ is less than 1. Given a path P from the start depot 0 to a node $i \in N$ we backtrack if $x^*(\widetilde{\Delta}^+(P)) \geqslant 1$, otherwise we extend the path to all $i \in N \setminus \{V(P)\}$. If an infeasible path is detected and it is minimal we have found a violated inequality; otherwise we backtrack again.

- **Odd CAT-inequalities:** We use the separation routine described in [10].
- D_k**-inequalities:** We use the separation routine described in [10].

Separation order Suppose we are considering a node in the branch-and-bound tree with the branching constraints $\underline{k} \leqslant x(N) \leqslant k$. The separation routines are called in the following order:

1. Subtour and π-inequalities (if $x(N) \leqslant 1$ we lift the π-inequalities)
2. σ-inequalities (if $x(N) \leqslant 1$ we lift the σ-inequalities)
3. If $x(N) \leqslant 1$ separation of (π, σ)-inequalities is performed
4. Tournament inequalities
5. S_1-inequalities
6. Odd CAT inequalities
7. D_k^+-inequalities
8. D_k^--inequalities

We skip all the subsequent routines whenever one of the procedures generate an inequality.

3.8.3 Branching

We have used two branching strategies: branching on the number of vehicles and branching on arcs:

- Branching on vehicles
  Given a point $x^* \in \mathbb{R}^A$ let $k = \lfloor x^*(N) \rfloor$. We create two subproblems if $x^*(N)$ is fractional: one by adding the constraint $x(N) \leqslant k$ and the other by adding the constraint $x(N) \geqslant k+1$.
- Branching on arcs
  If it is not possible to branch on the number of vehicles we branch on the x-variables. We use the default variable selection strategy of CPLEX [11].

3.8.4 Enumeration strategy

We have used best bound search as the enumeration strategy, which means that the node with the lowest objective function value will be selected.

3.9 Computational results

The DFJBC algorithm was tested on the class 2 Solomon test problems. The total number of test problems in this class is 81 when we include the test problems created by only considering the first 25 and 50 customers of each original 100 customer problem.

Key to Table 3.4: (preprocessing results)

ITER		:	Number of preprocessing loops
TW2 ...	#TW	:	Number of release time adjustments according to (3.32)
	TIME	:	Total increase in release times
TW3 ...	#TW	:	Number of release time adjustments according to (3.33)
	TIME	:	Total decrease in due times
REDUCTION		:	Total time of adjustments in percent; calculated by (TW2.TIME + TW3.TIME) $/\sum_{i \in N}(b_i - \max\{a_i, t_{0i}\})$

		TW2		TW3		
	ITER	#TW	TIME	#TW	TIME	REDUCTION
25 customers						
R101	4	7	32.0	3	21.8	21.5
C101	4	2	11.0	3	13.8	1.6
C106	4	2	17.5	2	21.2	2.1
RC101	3	1	9.8	0	0.0	1.3
R201	3	1	10.9	3	49.9	2.1
C201	11	14	940.1	16	2556.7	87.4
C205	3	1	320.0	2	0.8	4.0
C206	3	0	0.0	1	70.9	0.6
C208	3	0	0.0	1	47.2	0.3
RC201	3	0	0.0	1	17.9	0.6
50 customers						
R101	3	6	15.0	3	5.8	4.2
RC101	3	1	9.0	0	0.0	0.6
C201	6	7	5.0	8	4.7	0.1
C205	3	0	0.0	1	0.4	0.0
RC201	3	0	0.0	1	17.9	0.3
100 customers						
R101	3	3	8.9	1	7.0	1.6

Table 3.4: Preprocessing results

3.9.1 Preprocessing

First we present the results of the preprocessing routines. Only in 16 out of the 168 test problems do we observe adjustments according to criteria (3.32)-(3.33). The criterion (3.34) (or (3.30)) has no effect on the test problems. We do not include the adjustments of criterion (3.31) (or (3.29)); an additional 151 problems have adjustments only related to this criterion because of release times of zero. Note that there are also some minor adjustments due to the distance function we use. The number of remaining arc variables $|A|$ after preprocessing is presented in Section 3.9.2. The test problems would have $(n+1)n$ arcs in case the graphs were fully connected, i.e. the 25, 50, and 100 customer test problems would have 650, 2 550, and 10 100 arcs (variables), respectively. The number of remaining arcs in a test problem is the difference between the number of arcs in the complete graph and the number of precedence relationships $|R|$ induced by the time windows, i.e. $|A| = (n+1)n - |R|$.

3.9.2 Solomon's test problems

In this section we present the results for the class 2 Solomon test problems. Table 3.5 gives an overview of the results compared to algorithms in the literature. The algorithm of Chabrier [7] is an elementary shortest path decomposition of the VRPTW. In Irnich and Villeneuve [12] non-elementary paths are allowed but the formulation is strengthened by eliminating k-cycles ($k \geqslant 3$). In addition 1-path and 2-path inequalities are generated at the root node of the branch-and-bound tree. Kallehauge et al. [14] use a non-elementary shortest path with 2-cycle elimination but the master algorithm is accelerated by stabilizing the dual multipliers. Also, 2-path inequalities are generated in the root node and 1-path inequalities in all nodes of the branch-and-bound tree. It is clear that our cutting plane algorithm can be combined with all path pricing algorithms above. The question is where the relative advantages of the different algorithms lie. The results presented in this paper show that the polyhedral approach in certain cases outperform the shortest path decomposition of the VRPTW when the time windows are wide and therefore shows that the shortest path decomposition method is no longer the unchallenged winning strategy. Including arc pricing in our method would further improve the computational efficiency [2].

The new test problem reported solved in Table 3.5 by the DFJBC algorithm was R208.50. The solution consists of two routes with a total length of 487.7. In order to solve R208.50 we removed the 1 h CPU time limit. The solution time was 53815.2 s on the Sun Fire 15K. This brings the total number of solved test problems up to 63 out of 81. It is clear that the branch-and-cut code presented in this paper overall is not competitive, however, based on results on the 25-node problems the DFJBC algorithm shows that the polyhedral approach is promising for problems with wide time windows. We solved 44 problems within the 1 h CPU time limit, i.e. 24, 14, and 6 test problems out of the 27 25-node, 50-node, and 100-node test problems, respectively.

Now we compare our computational times for the class 2 Solomon test problems with Irnich and Villeneuve [12], Kallehauge et al. [14], and Chabrier [7]. Note that [7] only reports solution times for new solutions found and it is therefore only possible to compare solutions for 6 test problems.

Irnich and Villeneuve [12] performed all their computational experiments on a Pentium III 600MHz machine with 512Mb RAM. The Dell Inspiron 7500 machine is similar except that it only has 256Mb RAM. We should therefore be able to compare computational times for experiments performed on the Dell Inspiron 7500 with the results reported in Irnich and Villeneuve [12]. We tried to solve the 44 problems on the Dell Inspiron 7500 machine. However because of lack of memory we solved only 43 out of the 44 problems (RC203.25 was not solved). Irnich and Villeneuve [12] report solution times using 2-, 3-, and 4-cycle elimination. We compare our method with the k-cycle elimination method with the minimum solution time. In Table 3.6 we compare the total solution times per subclass and number of customers. In Table 3.7 we show the 10 problems for which we observed respectively the largest increase and decrease in the solution time by our method compared to the minimal solution time reported in Irnich and Villeneuve [12]. We describe the associated columns in Table 3.9.2. The results demonstrate that our method reduces the computational times significantly for a number of the 25-node problems, however, our method experienced problems with the R211.25 test problem solved by [12]. We will comment on this in Section 3.9.2. The DFJBC algorithm is also competitive with respect to the 50-node C-problems, however, for the remaining 50- and 100-node problems our method is not completive compared to [12]. Kallehauge et al. [14] also performed computational experiments on the Sun Fire 15K machine. In Table 3.8 and 3.9 we compare our results on the Sun Fire 15K machine with the results in [14]. Also in this comparison the DFJBC algorithm performs significantly better on the 25-node problems than the algorithm of [14]. The only exception is again the R211.25 test problem. Finally, in the comparison with Chabrier [7] in Table 3.10 we note that for the problems with randomly distributed customers and wide time windows (R208.25 and R204.25) the DFJBC algorithm performs well. However, it is also clear that our algorithm is not competitive in relation to the semi-clustered RC-problems nor 50- and 100-node problems. The results of Chabrier [7] has been obtained using a 1.5GHz Pentium IV with 256Mb RAM

Problem	C	IV	KLM	This paper	Total solved
Subclass R					
1	3	3	3	2	3
2	2	2	2	1	2
3	2	2	2	1	2
4	1	(1) 2	1	1	2
5	2	2	2	2	2
6	2	2	1	1	2
7	1	1	1	1	1
8	1	1	1	(1) 2	2
9	2	2	2	2	2
10	2	2	1	1	2
11	1	(1) 2	1	0	2
Total	19	(2) 21	17	14	22
Subclass C					
1	3	3	3	3	3
2	3	3	3	3	3
3	3	3	3	2	3
4	3	3	2	2	3
5	3	3	3	3	3
6	3	3	3	3	3
7	3	3	3	3	3
8	3	3	3	3	3
Total	24	24	23	22	24
Subclass RC					
1	3	3	3	2	3
2	3	3	1	1	3
3	2	2	0	1	2
4	(1) 1	0	0	0	1
5	3	3	1	2	3
6	2	2	1	2	2
7	(1) 2	1	0	1	2
8	(1) 1	0	0	0	1
Total	(3) 17	14	6	9	17
Grand total	(3) 60	(2) 59	46	(1) 45	63

Table 3.5: The number of Solomon problems solved in Chabrier [7] (C), Irnich and Villeneuve [12] (IV), Kallehauge et al. [14] (KLM), and this paper. 1 = 25 customer problem solved, 2 = 25- and 50-customer problem solved, 3 = 25-, 50-, and 100-customer problem solved. The numbers in parenthesis indicate solutions not reported elsewhere

Key to Table 3.7, 3.9, and 3.10

k	:	k-cycle elimination used in obtaining minimum solution time
RLB1	:	Value of root node before any strong valid inequalities are generated
RLB2	:	Value of root node after strong valid inequalities are generated
RLB	:	Value of root node
#N	:	Number of processed nodes in the branch-and-bound tree
CPU(TOTAL)	:	Total CPU-time in seconds to solve problem to optimality

Subclass	#P	IV	This paper
25 customers			
R	10	684.10	206.74
C	8	329.90	49.28
RC	5	1420.80	200.04
Total	23	2434.80	456.06
50 customers			
R	3	1353.20	5046.78
C	8	2136.90	1108.80
RC	3	1043.00	1462.51
Total	14	4533.10	7618.09
100 customers			
C	6	5973.80	6749.14
Total	6	5973.80	6749.14
Grand total	43	12941.70	14823.29

Table 3.6: Total solution time in seconds for the 43 class 2 test problems solved by the DFJBC algorithm on the Dell Inspiron 7500 vs. the corresponding minimum solution time reported by Irnich and Villeneuve [12] (IV)

and using the Java version of ILOG CPLEX 7.5. We have used the computational times obtained on the Dell Inspiron for comparison.

Results for 25-node problems

In this section we demonstrate in detail the performance of the branch-and-cut code on the class 2 Solomon test problems with 25 customer nodes. The results are summarized in Tables 3.11-3.14. Table 3.11 provides an overview of the results, Table 3.12 presents the number of generated cutting planes, Table 3.13 gives information on the branch-and-bound tree, and Table 3.14 summarizes the percentages of computing time spent in the various parts of the algorithm.

The branch-and-cut code shows promising results on the 25-node problems. However, three 25-node problems could not be solved within 1 h (R211.25, RC204.25, and RC208). For example the R204.25 and R208.25 problems are solved significantly faster by the branch-and-cut code than any other algorithm in the literature. It should be noted that these two problems have an optimal solution value close to the Hamilton path length of 312.3, see Table 3.1. This is not surprisingly an indication that the code will work well on problems where the capacity and time constraints are not very tight, i.e. with a solution structure close to the traveling salesman problem. It is interesting to note the column $|A|$ in Table 3.11. We can see that RC208.25 is a "pure" time-constrained problem, i.e. the time windows do not imply any precedence relationships. Also R211.25 has only 3 precedence relationships. These two problems are special because the time windows share a common band, which makes the problems very hard compared

	Irnich and Villeneuve [12]					This paper			
	k	RLB1	RLB2	#N	CPU(TOTAL)	RLB	#N	CPU(TOTAL)	DELTA
Top 10 increases									
R205.50	4	682.8	682.8	136	1091.40	662.5	2479	3722.51	2631.11
R209.50	4	599.8	599.8	6	255.40	582.4	420	1302.49	1047.09
C205.100	3	586.4	586.4	0	221.90	586.4	0	980.35	758.45
C206.100	3	586.0	586.0	0	814.40	586.0	0	1315.11	500.71
RC205.50	4	621.6	630.2	4	82.40	621.0	106	495.95	413.55
RC201.50	4	683.1	683.1	4	25.70	680.1	11	103.11	77.41
C202.100	2	589.1	589.1	0	585.60	589.1	0	659.13	73.53
R202.25	3	410.5	410.5	0	0.80	393.1	66	30.69	29.89
R203.25	4	391.4	391.4	0	5.80	373.0	111	28.38	22.58
R210.25	3	403.6	403.6	2	8.00	389.0	65	30.58	22.58
Top 10 decreases									
RC207.25	4	264.6	280.4	133	1393.70	268.3	2455	186.98	1206.72
C204.50	2	350.1	350.1	0	1159.40	347.8	70	648.20	511.20
R208.25	3	323.3	323.4	15	363.50	319.8	29	16.23	347.27
C208.100	2	581.8	585.8	1	2183.30	585.8	0	1884.58	298.72
C204.25	2	211.0	211.0	10	279.80	204.9	283	37.72	242.08
R204.25	4	349.1	349.1	34	231.70	332.2	83	29.70	202.00
C207.50	2	356.3	359.6	5	274.00	359.4	2	74.50	199.50
C207.100	2	582.0	585.8	4	2068.80	585.6	2	1887.45	181.35
C208.50	2	340.4	350.5	1	138.50	350.5	0	28.98	109.52
C202.50	2	360.2	360.2	0	196.80	354.1	12	92.00	104.80

Table 3.7: Top 10 increases and decreases in computational time using the DFJBC algorithm compared to Irnich and Villeneuve [19]

Subclass	#P	KLM	This paper
25 customers			
R	10	314.62	138.33
C	8	59.96	23.43
RC	4	424.84	13.49
Total	22	799.42	175.25
50 customers			
R	3	3089.91	2359.87
C	8	698.36	433.51
RC	1	3.04	42.30
Total	12	3791.31	2835.68
100 customers			
C	6	158.82	2065.03
Total	6	158.82	2065.03
Grand total	40	4749.55	5075.96

Table 3.8: Total solution time in seconds for the 40 class 2 test problems solved by both the LBCP algorithm in Kallehauge et al. [14] (KLM) and the DFJBC algorithm in this paper. All test problems are solved on the Sun Fire 15K machine

	Kallehauge et al. [14]				This paper			
	RLB1	RLB2	#N	CPU(TOTAL)	RLB	#N	CPU(TOTAL)	DELTA
Top 10 increases								
C208.100	581.8	585.8	0	57.02	585.8	0	569.74	512.72
C207.100	582.0	585.8	0	74.58	585.6	1	527.52	452.94
C206.100	585.4	586.0	0	14.90	586.0	0	418.99	404.09
C205.100	586.4	586.4	0	4.62	586.4	0	334.44	329.82
R209.50	582.9	588.4	524	530.41	582.4	626	816.31	285.90
C202.100	589.1	589.1	0	5.74	589.1	0	202.86	197.12
C202.50	360.2	360.2	0	20.34	354.1	13	62.55	42.21
RC201.50	670.2	682.0	14	3.04	680.1	8	42.30	39.26
R202.25	406.4	408.4	4	0.77	393.1	85	23.89	23.12
R203.25	379.9	381.6	36	5.41	373.0	111	24.13	18.72
Top 10 decreases								
R205.50	666.6	672.4	5254	2558.66	662.5	2320	1527.47	1031.19
RC202.25	290.4	313.4	664	222.25	311.8	56	10.13	212.12
RC206.25	250.1	289.0	502	195.49	324.0	0	1.55	193.94
C204.50	350.1	350.1	0	402.18	347.8	113	214.02	188.16
R204.25	333.1	335.4	776	190.64	332.2	68	17.60	173.04
C203.50	359.8	359.8	0	203.11	352.3	23	75.28	127.83
R208.25	318.1	318.9	74	58.23	319.9	17	7.33	50.90
C204.25	211.0	211.0	12	46.00	204.9	155	15.64	30.36
C206.50	344.2	359.0	4	33.88	359.8	0	15.22	18.66
R207.25	347.6	349.7	124	30.20	347.4	54	15.33	14.87

Table 3.9: Top 10 increases and decreases in computational time using the DFJBC algorithm compared to Kallehauge et al. [20]

	Chabrier [7]			This paper			
	RLB	#N	CPU(TOTAL)	RLB	#N	CPU(TOTAL)	DELTA
Increases							
R205.50	682.9	94	531.00	662.5	2479	3722.51	3191.51
R209.50	599.8	4	195.40	582.4	420	1302.49	1107.09
RC206.50	610.0	0	9.40	594.1	411	863.45	854.05
RC205.50	630.2	0	10.60	621.0	106	495.95	485.35
Decreases							
R208.25	328.2	0	741.50	319.8	29	16.23	725.27
R204.25	350.5	16	171.60	332.2	83	29.70	141.90

Table 3.10: Comparing the DFJBC algorithm to solutions reported in Chabrier [18]

Key to Table 3.11, 3.15, and 3.18:

$\|A\|$		:	Number of arcs after preprocessing
SOLUTION …	k	:	Number of vehicles in optimal solution
	OPT	:	Value of an optimal solution. If the instance is not solved to optimality, the global lower bound *glb* and global upper bound *gub* are given in the form $[glb, gub]$
	GAP	:	Optimality gap in percent; calculated by $(gub - glb)/glb \cdot 100$
ROOT …	BOUNDS	:	Lower bound *rlb* and upper bound *rub* at the root LP
	GAP	:	Optimality gap at the root node of the branch-and-cut (BC) tree in percent; calculated by $(rub - rlb)/rlb \cdot 100$
	QUAL	:	Quality of lower bound at the root of the BC tree in percent; calculated by $100 - (gub - rlb)/rlb \cdot 100$
BC-TREE …	#N	:	Number of processed nodes in the branch-and-cut tree
	LEV	:	Depth of the BC tree
#CUTS		:	Number of generated cutting planes
#LP		:	Number of linear programs solved
CPU		:	Total CPU-time in seconds to solve problem to optimality. If the problem could not be solved within a certain time limit we give the maximum CPU-time for computations before termination

Key to Table 3.12, 3.16, and 3.19:

SEC/ π	:	Number of generated SEC / Number of generated π inequalities / Number of calls of combined SEC and π separation routine
Lifted π	:	Number of generated strong π inequalities in BC-nodes where lower and upper bound on number of vehicles are both 1
σ	:	Number of generated σ inequalities / Number of calls of σ separation routine
Lifted σ	:	Number of generated strong σ inequalities in BC-nodes where lower and upper bound on number of vehicles are both 1
(π, σ)	:	Number of generated (π, σ) inequalities / Number of calls of (π, σ) separation routine
TOURN	:	Number of generated tournament constraints / Number of calls of tournament separation routine
S1	:	Number of generated S_1 inequalities / Number of calls of S_1 separation routine
CAT	:	Number of generated odd CAT inequalities / Number of calls of CAT separation routine
D_k^+	:	Number of generated D_k^+ inequalities / Number of calls of D_k^+ separation routine
D_k^-	:	Number of generated D_k^- inequalities / Number of calls of D_k^- separation routine

Key to Table 3.13:

#N	:	Total number of processed nodes / Total number of generated nodes
#NVEH	:	Number of processed vehicle branching nodes / Number of generated vehicle branching nodes
#NARC	:	Number of processed arc branching nodes / Number of generated arc branching nodes
k	:	Number of vehicles in optimal solution (or best incumbent when algorithm is terminated)
BC-TREE	:	In each generated interval of the lower and upper bound on the number of vehicles $[lbk, lbk]$ we show the total number of processed nodes vs. total number of generated nodes

Key to Table 3.14, 3.17, and 3.20:

INIT	:	Computing time spent in initialization phase (in %)
LP	:	Computing time spent in LP solver (in %)
SEP	:	Computing time spent in separation routines (in %)
MISC	:	Computing time spent in other parts of the program (in %)
TOTAL	:	Total CPU time in seconds

to R208.25 and R204.25 which both also have very few precedence relationships. Actually RC203.25 and RC204.25 have less precedence relationships than R208.25 but the semi-clustered distribution of the customers make these problems very hard. The conclusion is therefore that the code performs better on randomly distributed problems than semi-clustered problems and that problems with the special band-structure of time windows are much harder than problems where the time windows are non-overlapping.

Table 3.12 shows that the precedence constraints not surprisingly are important for problems with precedence constraints. In problems with little or no precedence structure the path inequalities are critical in achieving a tight lower bound (R211.25, RC208.25) and improvements are needed here. The lifted precedence constraints are seldomly generated and the CAT and D_k-constraints are mainly found in the R-subclass.

In our algorithm we simultaneously determine the number of vehicles and the design of the corresponding number of routes. In branch-and-cut algorithms for the CVRP it is standard to consider a fixed number of vehicles. The purpose of Table 3.13 is to provide an overview of the computational work of the separation of inequalities relative to the bounds on the number of vehicles. We note that for the class 2 problems with 25 customers a large proportion of the computational work is related to branch-and-bound nodes where the number of vehicles leaving the depot is one, i.e. we are in fact trying to solve an ATSPTW. We have taken advantage of this observation by implementing a separation routine for the (π,σ)-inequalities, but for the hard (and unsolved) instances further work is required in the case where we are considering one vehicle. We would also like to note that the optimal number of vehicles is always greater than one (except for R208.25) so the purpose of solving nodes with $x(N) = 1$ efficiently is being able to quickly prune that part of the search tree.

Table 3.14 gives the percentage of computing time spent in different parts of the algorithm. We can conclude that the majority of computing time is spent in the separation routines. However, for the hard problems (e.g. R211.25, RC204.25) a significant part of the computing time is spent in the LP solver. The reason for this is simply that as the number of generated cutting planes increases the LP problems become larger and more difficult.

Results for 50-node problems

In this section we present similar information as in the section with the 25-node results except we do not show details on the branch-and-bound tree. For the 50-node problems the algorithm is only effective in relation to the C-problems. However, the algorithm is overall an improvement compared to the algorithm of Kallehauge et al. [14] but especially compared to the algorithm of Chabrier [7] it is clear that the bounds are not tight enough for the R- and RC-problems compared to the bounds provided by the elementary shortest path decomposition. It is not possible to compare in details the results for the C-problems with [7], because the author does not provide computational times for this subclass. It is also clear that even for these relatively small problems the separation of subtour and precedence constraints with a complexity of $O(n^4)$ is a bottle-neck. Heuristic separation algorithms should be considered following the approach of e.g. Bard et al. [6].

Results for 100-node problems

In this section we provide similar information as in the section with the results for the 50-node problems. It is clear that the algorithm breaks down for the R- and RC-problems. In many cases it is not possible to finish the root node within the 1 h limit. For the 100-node problems the algorithm is also inefficient in relation to the C-problems compared to the algorithm of Kallehauge et al. [14]. However the bounds are still tight for the C-problems and therefore we believe it is possible to achieve an acceptable performance by introducing arc pricing.

		SOLUTION			ROOT			BC-TREE				
	$\|A\|$	k	OPT	GAP	BOUNDS	GAP	QUAL	#N	LEV	#CUTS	#LP	CPU
R201	397	4	463.3	0.00	463.3	0.00	100.00	0	0	11	24	0.64
R202	518	4	410.5	0.00	[393.1, 1244.6]	216.60	95.58	85	17	308	589	23.89
R203	596	3	391.4	0.00	[373.0, 1244.6]	233.69	95.06	111	21	347	691	24.13
R204	627	2	355.0	0.00	[332.2, 1244.6]	274.64	93.14	68	10	309	576	17.60
R205	488	3	393.0	0.00	[390.6, 393.0]	0.61	99.39	2	1	26	93	1.20
R206	565	3	374.4	0.00	[356.6, 405.4]	13.70	94.99	82	12	302	597	19.29
R207	609	3	361.6	0.00	[347.4, 392.9]	13.09	95.92	54	16	284	543	15.33
R208	631	1	328.2	0.00	[319.9, 336.9]	5.33	97.39	17	9	161	324	7.33
R209	545	2	370.7	0.00	[364.2, 372.6]	2.30	98.22	17	10	120	289	7.32
R210	560	3	404.6	0.00	[389.0, 448.4]	15.27	95.99	64	17	309	571	21.60
R211	647	2	[347.7, 352.7]	1.44	[312.7, 1244.6]	297.96	87.22	12792	255	6704	22279	__*
C201	353	2	214.7	0.00	214.7	0.00	100.00	0	0	0	1	0.15
C202	498	2	214.7	0.00	[209.8, 214.7]	2.36	97.64	2	1	22	52	1.04
C203	584	2	214.7	0.00	[209.8, 214.7]	2.36	97.64	2	1	54	158	2.12
C204	622	1	213.1	0.00	[204.9, 227.8]	11.19	95.98	155	22	279	605	15.64
C205	391	2	214.7	0.00	214.7	0.00	100.00	0	0	7	8	0.35
C206	415	2	214.7	0.00	214.7	0.00	100.00	0	0	22	47	0.94
C207	463	2	214.5	0.00	[214.3, 277.0]	29.26	99.91	1	1	42	90	1.81
C208	438	2	214.5	0.00	214.5	0.00	100.00	0	0	35	75	1.38
RC201	401	3	360.2	0.00	360.2	0.00	100.00	0	0	10	21	0.51
RC202	522	3	338.0	0.00	[311.8, 372.8]	19.56	91.60	56	29	181	403	10.13
RC203	596	3	326.9	0.00	[264.0, 986.3]	273.62	76.17	14498	68	4629	21250	1834.85
RC204	627	3	[276.7, 300.3]	8.54	[246.7, 368.1]	49.19	78.29	15101	81	13723	31972	__*
RC205	483	3	338.0	0.00	338	0.00	100.00	0	0	33	120	1.30
RC206	492	3	324.0	0.00	324	0.00	100.00	0	0	39	134	1.55
RC207	561	3	298.3	0.00	[268.3, 1884.4]	602.28	88.83	765	35	474	1593	50.55
RC208	650	3	[230.8, 334.7]	45.02	[226.5, 1884.4]	732.12	52.20	11856	327	8836	23865	__*

__* : time limit of 1 CPU hour exceeded

Table 3.11: Computational results for the DFJBC algorithm ($n = 25$)

Table 3.12: Number of generated cuts / Number of calls of separation routine ($n = 25$)

	SEC/ π	Lifted π	σ	Lifted σ	(π,σ)	TOURN	S1	CAT	D_k^+	D_k^-
R201	0/ 9/ 14	0/ 0	1/ 5	0/ 0	0/ 0	0/ 4	0/ 4	1/ 4	0/ 3	0/ 3
R202	2/ 129/ 379	5/ 49	85/ 191	0/ 3	52/ 69	10/ 106	16/ 96	9/ 80	0/ 71	0/ 71
R203	7/ 146/ 421	2/ 30	92/ 195	0/ 15	71/ 84	1/ 103	11/ 102	12/ 91	3/ 79	2/ 76
R204	16/ 141/ 362	0/ 28	58/ 148	0/ 17	57/ 83	1/ 90	13/ 89	13/ 76	7/ 63	3/ 56
R205	3/ 20/ 31	0/ 0	1/ 8	0/ 0	0/ 0	2/ 7	0/ 5	0/ 5	0/ 5	0/ 5
R206	5/ 108/ 369	2/ 22	105/ 221	0/ 0	33/ 35	6/ 116	10/ 110	25/ 100	6/ 75	2/ 69
R207	12/ 111/ 334	1/ 9	83/ 176	5/ 10	34/ 51	6/ 88	5/ 82	24/ 77	2/ 53	1/ 51
R208	20/ 76/ 183	0/ 3	30/ 85	0/ 0	2/ 2	0/ 55	5/ 55	16/ 50	8/ 34	4/ 26
R209	5/ 47/ 139	0/ 15	24/ 72	0/ 13	15/ 30	6/ 48	13/ 42	5/ 29	3/ 24	2/ 21
R210	9/ 122/ 371	1/ 39	71/ 181	0/ 12	58/ 90	8/ 110	20/ 102	17/ 82	1/ 65	2/ 64
R211	277/ 439/ 17650	0/ 166	398/ 16422	0/ 63	512/ 5171	1133/ 16024	3092/ 14891	640/ 11799	111/ 11159	102/ 11048
C201	0/ 0/ 1	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C202	1/ 16/ 26	0/ 0	5/ 9	0/ 0	0/ 0	0/ 4	0/ 4	0/ 4	0/ 4	0/ 4
C203	7/ 40/ 61	0/ 1	5/ 13	0/ 0	1/ 1	1/ 8	0/ 7	0/ 7	0/ 7	0/ 7
C204	33/ 84/ 386	0/ 17	29/ 266	0/ 0	3/ 43	80/ 237	18/ 157	13/ 139	10/ 126	9/ 116
C205	0/ 7/ 8	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C206	0/ 22/ 25	0/ 0	0/ 3	0/ 0	0/ 0	0/ 3	0/ 3	0/ 3	0/ 3	0/ 3
C207	0/ 30/ 48	0/ 0	0/ 18	0/ 0	0/ 0	9/ 18	2/ 9	0/ 7	0/ 7	1/ 7
C208	2/ 31/ 38	0/ 0	1/ 5	0/ 0	0/ 0	1/ 4	0/ 3	0/ 3	0/ 3	0/ 3
RC201	0/ 9/ 12	0/ 0	1/ 3	0/ 0	0/ 0	0/ 2	0/ 2	0/ 2	0/ 2	0/ 2
RC202	3/ 92/ 229	2/ 29	49/ 125	0/ 0	7/ 7	4/ 76	16/ 72	5/ 56	2/ 51	1/ 49
RC203	27/ 1238/ 12810	65/ 1563	446/ 8896	0/ 9	2584/ 2590	2/ 8450	141/ 8448	101/ 8307	11/ 8206	14/ 8195
RC204	89/ 3008/ 27272	324/ 4911	1242/ 15447	0/ 71	8404/ 8762	28/ 14205	370/ 14177	230/ 13807	14/ 13577	14/ 13563
RC205	2/ 25/ 34	0/ 0	6/ 7	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
RC206	5/ 30/ 41	0/ 0	4/ 6	0/ 0	0/ 0	0/ 2	0/ 2	0/ 2	0/ 2	0/ 2
RC207	23/ 155/ 913	3/ 41	131/ 692	1/ 13	40/ 60	18/ 560	34/ 542	43/ 508	14/ 465	12/ 451
RC208	536/ 0/ 20687	0/ 0	0/ 20151	0/ 0	0/ 2658	2806/ 20151	4376/ 17345	850/ 12969	134/ 12119	134/ 11985

Table 3.13: Number of processed branch-and-bound nodes / Number of generated nodes ($n = 25$)

	#N	#NVEH	#NARC	k	BC-TREE					
R202	85/ 134	9/ 12	76/ 122	4	[1,1] 1 / 3	[2,2] 33 / 51	[2,25] 16 / 19	[3,3] 17 / 30	[3,25] 17 / 27	[4,25] 1 / 4
R203	111/ 144	8/ 8	103/ 136	3	[1,1] 1 / 3	[2,2] 43 / 61	[2,25] 56 / 67	[3,25] 11 / 13	0	0
R204	68/ 96	2/ 2	65/ 94	2	[1,1] 9 / 13	[2,25] 58 / 83	0	0	0	0
R205	2/ 2	0/ 0	2/ 2	3	[1,25] 2 / 2	0	0	0	0	0
R206	82/ 124	10/ 10	72/ 114	3	[1,1] 1 / 3	[2,2] 13 / 22	[2,25] 55 / 81	[3,25] 13 / 18	0	0
R207	54/ 84	2/ 2	52/ 82	3	[1,1] 1 / 3	[2,25] 53 / 81	0	0	0	0
R208	17/ 28	2/ 2	15/ 26	1	[1,1] 1 / 1	[1,25] 14 / 22	[2,25] 2 / 5	0	0	0
R209	17/ 22	2/ 2	14/ 20	2	[1,1] 1 / 1	[2,25] 15 / 21	0	0	0	0
R210	64/ 114	9/ 14	55/ 100	3	[1,1] 1 / 3	[2,2] 14 / 24	[2,25] 47 / 79	[3,25] 2 / 8	0	0
R211	12792/ 21880	304/ 656	12488/ 21224	2	[1,1] 2000 / 3394	[1,25] 10607 / 17922	[2,25] 185 / 564	0	0	0
C202	2/ 2	2/ 2	0/ 0	2	[1,1] 1 / 1	[2,25] 1 / 1	0	0	0	0
C203	2/ 2	2/ 2	0/ 0	2	[1,1] 1 / 1	[2,25] 1 / 1	0	0	0	0
C204	155/ 204	11/ 12	144/ 192	1	[1,1] 25 / 32	[1,25] 125 / 166	[2,25] 5 / 6	0	0	0
C207	1/ 4	0/ 0	1/ 4	2	[1,25] 1 / 4	0	0	0	0	0
RC202	56/ 86	7/ 8	49/ 78	3	[1,1] 1 / 1	[2,2] 36 / 55	[2,25] 17 / 27	[3,25] 2 / 3	0	0
RC203	14498/ 16326	1547/ 1826	12951/ 14500	3	[1,1] 463 / 486	[1,2] 1 / 1	[1,25] 1211 / 1238	[2,2] 685 / 924	[2,25] 11848 / 13238	[3,25] 290 / 439
RC204	15101/ 27076	1632/ 6496	13469/ 20580	3	[1,1] 925 / 3413	[1,25] 12060 / 16350	[2,2] 30 / 95	[2,25] 2083 / 7177	[3,25] 3 / 41	0
RC207	765/ 870	40/ 40	725/ 830	3	[1,1] 2 / 4	[1,25] 2 / 2	[2,2] 156 / 196	[2,25] 587 / 644	[3,25] 18 / 24	0
RC208	11856/ 23692	21/ 42	11835/ 23650	3	[1,1] 947 / 1905	[1,25] 10909 / 21766	[2,25] 0 / 21	0	0	0

	INIT	LP	SEP	MISC	TOTAL
R201	14.1	4.7	73.4	7.8	0.64
R202	0.3	6.7	83.6	9.3	23.89
R203	0.3	9.4	70.2	20.1	24.13
R204	0.5	13.4	69.6	16.5	17.60
R205	6.7	8.3	78.3	6.7	1.20
R206	0.3	6.9	78.7	14.1	19.29
R207	0.5	8.3	74.2	17.0	15.33
R208	1.1	4.9	79.7	14.3	7.33
R209	1.0	3.4	86.3	9.3	7.32
R210	0.3	7.0	78.9	13.8	21.60
R211	0.0	58.2	39.1	2.7	3652.21
C201	53.3	0.0	33.3	13.3	0.15
C202	6.7	5.8	76.9	10.6	1.04
C203	3.8	5.2	82.1	9.0	2.12
C204	0.4	6.2	86.1	7.3	15.64
C205	20.0	0.0	74.3	5.7	0.35
C206	7.4	4.3	83.0	5.3	0.94
C207	3.9	3.9	85.6	6.6	1.81
C208	5.1	2.9	86.2	5.8	1.38
RC201	15.7	2.0	74.5	7.8	0.51
RC202	0.7	4.7	86.1	8.5	10.13
RC203	0.0	63.5	34.7	1.8	1834.85
RC204	0.0	60.0	37.4	2.6	3691.98
RC205	6.2	5.4	77.7	10.8	1.30
RC206	4.5	4.5	79.4	11.6	1.55
RC207	0.1	13.2	79.5	7.1	50.55
RC208	0.0	50.3	46.2	3.5	3643.42

Table 3.14: Percentage of computing time spent in different parts of the DFJBC algorithm ($n = 25$)

		SOLUTION			ROOT			BC-TREE				
	$\|A\|$	k	OPT	GAP	BOUNDS	GAP	QUAL	#N	LEV	#CUTS	#LP	CPU
R201	1518	6	791.9	0.00	791.9	0.00	100.00	0	0	32	90	16.09
R202	1984	50	[638.8, 2620.8]	310.25	[635.8, 2620.8]	312.24	-212.24	1927	617	2879	6362	__*
R203	2295	50	[552.5, 2620.8]	374.35	[541.1, 2620.8]	384.31	-284.31	1893	507	3477	6698	__*
R204	2514	4	[483.5, 543.2]	12.34	[471.8, 2620.8]	455.48	84.87	2570	921	3604	7548	__*
R205	1878	4	690.1	0.00	[662.5, 707.6]	6.81	95.83	2320	32	772	4502	1527.47
R206	2183	4	[596.4, 722.4]	21.13	[572.1, 2620.8]	358.09	73.73	2209	510	3007	7310	__*
R207	2380	3	[532.8, 610.1]	14.52	[520.9, 2620.8]	403.13	82.88	2419	873	3199	6846	__*
R208	2525	3	[471.6, 534.4]	13.31	[464.1, 2620.8]	464.65	84.86	3514	500	3070	8167	__*
R209	2150	4	600.6	0.00	[582.4, 2620.8]	349.98	96.88	626	124	883	2253	816.31
R210	2188	4	[594.0, 670.9]	12.95	[582.7, 2620.8]	349.79	84.86	2001	517	3221	6592	__*
R211	2523	50	[467.8, 2620.8]	460.26	[464.6, 2620.8]	464.12	-364.12	2466	854	3351	6962	__*
C201	1339	3	360.2	0.00	360.2	0.00	100.00	0	0	0	1	0.83
C202	1890	3	360.2	0.00	[354.1, 382.1]	7.92	98.26	13	6	117	174	62.55
C203	2253	3	359.8	0.00	[352.3, 359.8]	2.14	97.86	23	9	159	354	75.28
C204	2505	2	350.1	0.00	[347.8, 363.1]	4.40	99.34	113	63	411	759	214.02
C205	1476	3	359.8	0.00	359.8	0.00	100.00	0	0	22	65	11.11
C206	1589	3	359.8	0.00	359.8	0.00	100.00	0	0	33	118	15.22
C207	1814	3	359.6	0.00	[359.4, 366.2]	1.89	99.94	2	2	75	208	37.48
C208	1691	2	350.5	0.00	350.5	0.00	100.00	0	0	34	129	17.02
RC201	1495	5	684.8	0.00	[680.1, 775.6]	14.05	99.30	8	3	67	166	42.30
RC202	1971	5	[591.2, 621.9]	5.20	[515.2, 872.8]	69.42	79.28	2657	265	2399	6536	__*
RC203	2291	50	[480.2, 4056.0]	744.65	[446.4, 4056.0]	808.52	-708.52	2168	792	3156	6634	__*
RC204	2513	3	[396.9, 499.3]	25.80	[389.6, 4056.0]	940.97	71.86	3958	790	2713	8250	__*
RC205	1834	5	630.2	0.00	[621.0, 738.6]	18.94	98.51	90	13	212	547	149.71
RC206	1860	5	610.0	0.00	[594.1, 4056.0]	582.69	97.33	470	22	298	1151	329.28
RC207	2173	50	[500.7, 4056.0]	710.09	[495.2, 4056.0]	719.11	-619.11	2459	963	2268	6062	__*
RC208	2548	50	[373.6, 4056.0]	985.76	[372.4, 4056.0]	989.07	-889.07	3079	785	3161	7373	__*

__* : time limit of 1 CPU hour exceeded

Table 3.15: Computational results for the DFJBC algorithm ($n = 50$)

Table 3.16: Number of generated cuts / Number of calls of separation routine ($n = 50$)

	SEC/ π	Lifted π	σ	Lifted σ	(π, σ)	TOURN	S1	CAT	D_k^+	D_k^-
R201	2/ 24/ 34	0/ 0	5/ 8	0/ 0	0/ 0	1/ 3	0/ 2	0/ 2	0/ 2	0/ 2
R202	7/ 1407/ 4802	25/ 786	974/ 3101	2/ 25	262/ 290	32/ 2125	120/ 2093	37/ 1973	9/ 1936	4/ 1927
R203	15/ 1819/ 5358	13/ 486	1286/ 3306	0/ 13	205/ 221	32/ 2020	65/ 1988	26/ 1923	10/ 1897	6/ 1887
R204	106/ 1421/ 6173	31/ 545	893/ 3921	0/ 57	694/ 803	16/ 3028	38/ 3012	318/ 2974	50/ 2656	37/ 2606
R205	9/ 230/ 2173	0/ 0	141/ 1934	0/ 0	0/ 0	69/ 1793	166/ 1724	130/ 1558	17/ 1428	10/ 1411
R206	17/ 1265/ 5210	8/ 170	1108/ 3828	0/ 10	92/ 108	62/ 2720	310/ 2658	116/ 2348	18/ 2232	11/ 2214
R207	47/ 1682/ 5609	9/ 68	1034/ 3820	0/ 12	51/ 59	67/ 2786	196/ 2719	81/ 2523	19/ 2442	13/ 2423
R208	175/ 1146/ 6583	6/ 448	502/ 4833	3/ 89	423/ 538	121/ 4328	149/ 4207	439/ 4058	58/ 3619	48/ 3561
R209	23/ 355/ 1295	0/ 0	226/ 917	0/ 0	0/ 0	44/ 691	130/ 647	70/ 517	18/ 447	17/ 429
R210	21/ 1544/ 5215	5/ 108	887/ 3508	1/ 48	137/ 162	132/ 2620	421/ 2488	63/ 2067	6/ 2004	4/ 1998
R211	120/ 736/ 5811	394/ 1297	301/ 4414	288/ 560	147/ 4940	365/ 3825	790/ 3460	131/ 2670	52/ 2539	27/ 2487
C201	0/ 0/ 1	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C202	2/ 68/ 131	0/ 0	28/ 61	0/ 0	0/ 0	6/ 33	11/ 27	0/ 16	2/ 16	0/ 14
C203	14/ 101/ 178	0/ 0	34/ 63	0/ 0	0/ 0	3/ 29	6/ 26	1/ 20	0/ 19	0/ 19
C204	37/ 137/ 515	0/ 0	87/ 341	0/ 0	0/ 0	39/ 254	30/ 215	39/ 185	24/ 146	18/ 122
C205	0/ 21/ 24	0/ 0	0/ 3	0/ 0	0/ 0	1/ 3	0/ 2	0/ 2	0/ 2	0/ 2
C206	4/ 29/ 34	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C207	3/ 62/ 83	0/ 0	1/ 18	0/ 0	0/ 0	4/ 17	2/ 13	2/ 11	1/ 9	0/ 8
C208	4/ 27/ 37	0/ 0	3/ 6	0/ 0	0/ 0	0/ 3	0/ 3	0/ 3	0/ 3	0/ 3
RC201	2/ 40/ 83	0/ 0	2/ 41	0/ 0	0/ 0	12/ 39	8/ 27	3/ 19	0/ 16	0/ 16
RC202	16/ 1112/ 4916	36/ 824	762/ 3692	0/ 0	60/ 60	28/ 2930	342/ 2902	26/ 2560	10/ 2534	7/ 2524
RC203	32/ 1526/ 5319	28/ 1025	819/ 3309	0/ 5	424/ 435	61/ 2490	168/ 2429	65/ 2261	22/ 2196	11/ 2174
RC204	117/ 1004/ 6672	22/ 371	638/ 5103	3/ 72	426/ 477	46/ 4462	118/ 4416	244/ 4298	56/ 4054	39/ 3998
RC205	6/ 112/ 292	0/ 0	43/ 174	0/ 0	0/ 0	24/ 131	20/ 107	6/ 87	0/ 81	1/ 81
RC206	15/ 126/ 586	0/ 0	77/ 445	0/ 0	0/ 0	12/ 368	34/ 356	22/ 322	5/ 300	7/ 295
RC207	22/ 761/ 4724	15/ 169	651/ 3563	3/ 45	363/ 808	34/ 2909	299/ 2875	67/ 2576	29/ 2509	24/ 2480
RC208	308/ 169/ 6228	31/ 184	121/ 5602	17/ 85	118/ 6670	744/ 5464	1344/ 4720	181/ 3376	83/ 3195	45/ 3112

	INIT	LP	SEP	MISC	TOTAL
R201	1.0	1.5	96.0	1.6	16.09
R202	0.0	5.2	89.2	5.6	3710.71
R203	0.0	11.4	79.3	9.3	3675.22
R204	0.0	15.9	76.5	7.5	3660.86
R205	0.0	10.6	86.9	2.5	1527.47
R206	0.0	13.4	81.9	4.7	3672.94
R207	0.0	10.7	77.7	11.6	3657.81
R208	0.0	12.4	83.6	4.0	3662.96
R209	0.0	9.3	85.3	5.3	816.31
R210	0.0	9.3	81.9	8.8	3678.79
R211	0.0	8.4	81.5	10.1	3709.52
C201	20.5	1.2	72.3	6.0	0.83
C202	0.3	0.7	97.5	1.5	62.55
C203	0.2	1.2	96.2	2.3	75.28
C204	0.1	2.0	94.5	3.4	214.02
C205	1.6	1.0	95.3	2.1	11.11
C206	1.1	1.2	95.0	2.8	15.22
C207	0.4	1.1	96.7	1.7	37.48
C208	1.0	1.1	95.4	2.5	17.02
RC201	0.4	0.7	97.5	1.3	42.30
RC202	0.0	17.3	80.4	2.4	3647.85
RC203	0.0	7.5	84.1	8.4	3682.08
RC204	0.0	9.0	84.7	6.3	3653.89
RC205	0.1	1.2	97.0	1.7	149.71
RC206	0.1	2.0	95.5	2.4	329.28
RC207	0.0	4.5	89.5	6.0	3759.56
RC208	0.0	9.8	82.0	8.2	3639.20

Table 3.17: Percentage of computing time spent in different parts of the DFJBC algorithm ($n = 50$)

		SOLUTION			ROOT			BC-TREE				
	$\|A\|$	k	OPT	GAP	BOUNDS	GAP	QUAL	#N	LEV	#CUTS	#LP	CPU
R201	5917	9	[1132.7, 1155.6]	2.03	[1123.6, 4980.0]	343.23	97.15	72	26	295	666	__*
R202	7718	100	[888.6, 4980.0]	460.42	[888.6, 4980.0]	460.42	-360.42	0	0	464	1165	__*
R203	9050	100	[748.1, 4980.0]	565.70	[748.1, 4980.0]	565.70	-465.70	0	0	538	1219	__*
R204	9845	100	[661.9, 4980.0]	652.39	[661.9, 4980.0]	652.41	-552.41	14	11	563	1070	__*
R205	7327	100	[900.0, 4980.0]	453.33	[899.7, 4980.0]	453.53	-353.53	7	4	402	865	__*
R206	8521	100	[783.6, 4980.0]	535.52	[783.6, 4980.0]	535.53	-435.53	2	2	463	1108	__*
R207	9378	100	[714.8, 4980.0]	596.69	[714.8, 4980.0]	596.70	-496.70	2	2	543	1231	__*
R208	9936	100	[651.8, 4980.0]	664.02	[651.6, 4980.0]	664.29	-564.29	21	16	571	1065	__*
R209	8518	100	[785.8, 4980.0]	533.79	[785.2, 4980.0]	534.22	-434.22	8	5	476	1010	__*
R210	8568	100	[798.3, 4980.0]	523.85	[798.2, 4980.0]	523.87	-423.87	5	4	456	1074	__*
R211	9997	100	[645.1, 4980.0]	671.99	[645.1, 4980.0]	672.01	-572.01	37	31	553	958	__*
C201	5221	3	589.1	0.00	589.1	0.00	100.00	0	0	0	1	11.48
C202	7350	3	589.1	0.00	589.1	0.00	100.00	0	0	26	70	202.86
C203	8866	4	[586.0, 632.3]	7.90	[584.4, 643.8]	10.17	91.79	66	19	447	821	__*
C204	9789	3	[584.4, 597.1]	2.18	[583.5, 5934.4]	917.06	97.67	95	47	509	929	__*
C205	5698	3	586.4	0.00	586.4	0.00	100.00	0	0	38	85	334.44
C206	6212	3	586.0	0.00	586	0.00	100.00	0	0	50	158	418.99
C207	6578	3	585.8	0.00	[585.6, 585.8]	0.03	99.97	1	1	57	212	527.52
C208	6665	3	585.8	0.00	585.8	0.00	100.00	0	0	65	206	569.74
RC201	5918	7	[1250.1, 1288.2]	3.05	[1249.2, 6609.4]	429.09	96.88	61	48	309	693	__*
RC202	7752	100	[940.1, 6609.4]	603.03	[940.1, 6609.4]	603.03	-503.03	0	0	467	1071	__*
RC203	9056	100	[781.6, 6609.4]	745.67	[781.6, 6609.4]	745.67	-645.67	1	1	522	1239	__*
RC204	9854	100	[692.7, 6609.4]	854.19	[692.7, 6609.4]	854.21	-754.21	18	13	562	1043	__*
RC205	7173	100	[1081.7, 6609.4]	511.00	[1081.5, 6609.4]	511.12	-411.12	7	5	414	893	__*
RC206	7366	100	[974.8, 6609.4]	578.01	[974.8, 6609.4]	578.01	-478.01	1	1	416	955	__*
RC207	8619	100	[832.4, 6609.4]	694.03	[832.4, 6609.4]	694.03	-594.03	0	0	480	1095	__*
RC208	10091	100	[647.7, 6609.4]	920.43	[647.7, 6609.4]	920.44	-820.44	226	221	372	907	__*

__* : time limit of 1 CPU hour exceeded

Table 3.18: Computational results for the DFJBC algorithm ($n = 100$)

Table 3.19: Number of generated cuts / Number of calls of separation routine ($n = 100$)

	SEC/ π	Lifted π	σ	Lifted σ	(π, σ)	TOURN	S1	CAT	D_k^+	D_k^-
R201	5/ 120/ 367	0/ 0	60/ 242	0/ 0	0/ 0	35/ 182	37/ 147	34/ 110	3/ 76	1/ 73
R202	5/ 383/ 464	0/ 0	76/ 76	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0
R203	13/ 439/ 538	0/ 0	86/ 86	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0
R204	33/ 357/ 578	0/ 0	127/ 188	0/ 0	0/ 0	3/ 61	8/ 58	18/ 50	9/ 32	8/ 23
R205	14/ 221/ 411	0/ 0	104/ 176	0/ 0	0/ 0	5/ 72	33/ 67	21/ 34	2/ 13	2/ 11
R206	9/ 314/ 467	0/ 0	118/ 144	0/ 0	0/ 0	2/ 26	9/ 24	7/ 15	1/ 8	3/ 7
R207	23/ 369/ 547	0/ 0	126/ 155	0/ 0	0/ 0	0/ 29	7/ 29	11/ 22	5/ 11	2/ 6
R208	50/ 294/ 594	0/ 0	143/ 250	0/ 0	0/ 0	0/ 107	7/ 107	52/ 100	19/ 48	6/ 29
R209	22/ 251/ 486	0/ 0	171/ 213	0/ 0	0/ 0	5/ 42	20/ 37	6/ 17	0/ 11	1/ 11
R210	9/ 273/ 463	0/ 0	156/ 181	0/ 0	0/ 0	2/ 25	12/ 23	4/ 11	0/ 7	0/ 7
R211	62/ 223/ 592	0/ 0	79/ 307	0/ 0	0/ 0	20/ 228	77/ 208	48/ 131	26/ 83	18/ 57
C201	0/ 0/ 1	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C202	0/ 26/ 27	0/ 0	0/ 1	0/ 0	0/ 0	0/ 1	0/ 1	0/ 1	0/ 1	0/ 1
C203	28/ 228/ 509	0/ 0	44/ 253	0/ 0	0/ 0	98/ 209	37/ 111	8/ 74	2/ 66	2/ 64
C204	53/ 225/ 603	0/ 0	60/ 325	0/ 0	0/ 0	73/ 265	52/ 192	23/ 140	17/ 117	6/ 100
C205	1/ 35/ 41	0/ 0	0/ 5	0/ 0	0/ 0	1/ 5	1/ 4	0/ 3	0/ 3	0/ 3
C206	5/ 42/ 52	0/ 0	2/ 5	0/ 0	0/ 0	0/ 3	1/ 3	0/ 2	0/ 2	0/ 2
C207	1/ 47/ 63	0/ 0	3/ 15	0/ 0	0/ 0	2/ 12	3/ 10	0/ 7	0/ 7	1/ 7
C208	9/ 48/ 70	0/ 0	3/ 13	0/ 0	0/ 0	2/ 10	1/ 8	2/ 7	0/ 5	0/ 5
RC201	3/ 138/ 372	0/ 0	51/ 231	0/ 0	0/ 0	35/ 180	47/ 145	26/ 98	5/ 72	4/ 67
RC202	12/ 373/ 467	0/ 0	82/ 82	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0
RC203	12/ 371/ 525	0/ 0	115/ 142	0/ 0	0/ 0	2/ 27	14/ 25	3/ 11	1/ 8	4/ 7
RC204	30/ 314/ 582	0/ 0	150/ 238	0/ 0	0/ 0	1/ 88	28/ 87	21/ 59	9/ 38	9/ 29
RC205	2/ 263/ 423	0/ 0	103/ 158	0/ 0	0/ 0	5/ 55	34/ 50	4/ 16	3/ 12	0/ 9
RC206	17/ 243/ 419	0/ 0	98/ 159	0/ 0	0/ 0	7/ 61	45/ 54	2/ 9	4/ 7	0/ 3
RC207	16/ 303/ 481	0/ 0	134/ 162	0/ 0	0/ 0	3/ 28	15/ 25	3/ 10	2/ 7	4/ 5
RC208	115/ 43/ 600	0/ 0	25/ 442	0/ 0	0/ 0	23/ 417	84/ 394	41/ 310	22/ 269	19/ 247

	INIT	LP	SEP	MISC	TOTAL
R201	0.0	0.2	99.1	0.7	3616.18
R202	0.0	0.9	98.6	0.5	3615.42
R203	0.0	1.6	97.8	0.6	3623.32
R204	0.0	1.9	93.0	5.0	3616.80
R205	0.0	0.7	96.1	3.3	3655.33
R206	0.0	1.2	94.1	4.7	3655.12
R207	0.0	1.9	92.5	5.6	3736.64
R208	0.0	1.6	93.9	4.5	3635.19
R209	0.0	1.0	95.1	3.9	3872.42
R210	0.0	1.2	94.2	4.6	3650.14
R211	0.0	1.5	94.3	4.2	3621.49
C201	4.4	0.3	94.3	1.0	11.48
C202	0.3	0.3	99.0	0.5	202.86
C203	0.0	0.3	99.2	0.5	3605.29
C204	0.0	0.6	98.1	1.3	3605.84
C205	0.2	0.2	99.4	0.3	334.44
C206	0.1	0.2	99.1	0.5	418.99
C207	0.1	0.2	99.1	0.6	527.52
C208	0.1	0.2	99.2	0.5	569.74
RC201	0.0	0.2	99.1	0.7	3617.12
RC202	0.0	0.9	98.7	0.4	3614.09
RC203	0.0	1.7	93.4	4.9	3740.49
RC204	0.0	1.8	92.8	5.3	3634.98
RC205	0.0	0.5	97.3	2.2	3665.94
RC206	0.0	0.7	96.4	2.9	3626.73
RC207	0.0	1.2	96.5	2.2	3615.31
RC208	0.0	0.8	95.9	3.4	3614.08

Table 3.20: Percentage of computing time spent in different parts of the DFJBC algorithm ($n = 100$)

3.10 Conclusions

In this paper we presented a new formulation of the VRPTW involving only binary arc variables. The new formulation is based on the formulation of the ATSPTW by Ascheuer et al. [3] and has the advantage of avoiding additional variables and linking constraints. In the new formulation of the VRPTW time windows are modeled using path inequalities. A path inequality eliminates a path that is infeasible because of some deadline or vehicle capacity is violated. We presented a new class of strengthened path inequalities based on the polyhedral results obtained by Mak [17] in the context of the TSP with replenishment arcs. We studied the VRPTW polytope and determined the polytope dimension. We shoved that the new class of path inequalities is facet defining under reasonable assumptions. These are the first polyhedral results for the VRPTW. We introduced precedence constraints in the context of the VRPTW. We designed a branch-and-cut algorithm for the exact solution of the VRPTW and evaluated the computational performance on the long-horizon Solomon test problems. The outcome is based on 25-node problems that the algorithm shows promising results compared to leading algorithms in the literature. In particular we report a solution to a previously unsolved 50-node Solomon test problem R208. The conclusion is therefore that the path pricing algorithm is no longer the unchallenged winning strategy for solving the VRPTW.

Acknowledgements We would like to thank Dr. Vicky Mak of Deakin University, Australia, for all the fruitful discussions regarding this paper. We would also like to thank Dr. Irina Dumitrescu of HEC Montreal, Canada, for giving corrections and suggestions improving this paper. The computational platform for carrying out the tests on the Sun Fire 15K machine were provided by The Danish Center for Scientific Computing.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, New Jersey, 1993. [71]

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Mathematical Programming Series B*, 97:91–153, 2003. [71, 74]

[3] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36:69–79, 2000. [57, 58, 59, 62, 91]

[4] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming Series A*, 90:475–506, 2001. [58, 59, 65, 68, 71]

[5] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995. [58, 66, 71]

[6] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002. [58, 59, 65, 80]

[7] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. Technical report, ILOG, 2003. [74, 75, 78, 80]

[8] V. Chvátal. *Linear Programming*. Freeman, New York, 1983. [71]

[9] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992. [57, 68]

[10] M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43:1520–1536, 1997. [67, 72]

[11] ILOG. *ILOG CPLEX 9.1 User's Manual*, 2005. [69, 71, 72]

[12] S. Irnich and D. Villeneuve. The shortest path problem with k-cycle elimination ($k \geqslant 3$): Improving a branch-and-price algorithm for the VRPTW. Technical report, Lehr- und Forschungsgebiet Operations Research und Logistik Management, Rheinisch-Westfälische Technische Hochschule, 2003. [74, 75, 76, 77]

[13] M. Jünger and S. Thienel. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software: Practice and Experience*, 30:1325–1352, 2000. [71]

[14] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33:1464–1487, 2006. [74, 75, 77, 78, 80]

[15] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999. [67]

[16] G. Kontoravdis and J. F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995. [68]

[17] V. H. Mak. *On the Asymmetric Travelling Salesman Problem with Replenishment Arcs*. PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, 2001. [57, 58, 60, 61, 91]

[18] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960. [58]

[19] M. W. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985. [58]

[20] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–65, 1987. [59, 68]

[21] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998. [71]

Chapter 4

Vehicle routing problem with time windows

Brian Kallehauge
Centre for Traffic and Transport, Technical University of Denmark

Jesper Larsen
Informatics and Mathematical Modelling, Technical University of Denmark

Oli B. G. Madsen
Centre for Traffic and Transport, Technical University of Denmark

Marius M. Solomon
College of Business Administration, Northeastern University and GERAD

Abstract

In this chapter we discuss the vehicle routing problem with time windows in terms of its mathematical modeling, its structure and decomposition alternatives. We then present the master problem and the subproblem for the column generation approach, respectively. Next, we illustrate a branch-and-bound framework and address acceleration strategies used to increase the efficiency of branch-and-price methods. Then, we describe generalizations of the problem and report computational results for the classic Solomon test sets. Finally, we present our conclusions and discuss some open problems.

4.1 Introduction

The vehicle routing problem (VRP) involves finding a set of routes, starting and ending at a depot, that together cover a set of customers. Each customer has a given demand, and no vehicle can service more customers than its capacity permits. The objective is to minimize the total distance traveled or the number of vehicles used, or a combination of these. In this chapter, we consider the vehicle routing problem with time windows (VRPTW), which is a generalization of the VRP where the service at any customer starts within a given time interval, called a time window. Time windows are called soft when they can be considered non-biding for a penalty cost. They are hard when they cannot be violated, i.e., if a vehicle

arrives too early at a customer, it must wait until the time window opens; and it is not allowed to arrive late. This is the case we consider here.

The remarkable advances in information technology have enabled companies to focus on efficiency and timeliness throughout the supply chain. In turn, the VRPTW has increasingly become an invaluable tool in modeling a variety of aspects of supply chain design and operation. Important VRPTW applications include deliveries to supermarkets, bank and postal deliveries, industrial refuse collection, school bus routing, security patrol service, and urban newspaper distribution. Its increased practical visibility has evolved in parallel with the development of broader and deeper research directed at its solution. Significant progress has been made in both the design of heuristics and the development of optimal approaches.

In this chapter we will concentrate on exact methods for the VRPTW based on column generation. These date back to Desrochers, Desrosiers, and Solomon [12] who used column generation in a Dantzig-Wolfe decomposition framework and Halse [20] who implemented a decomposition based on variable splitting (also known as Lagrangean decomposition). Later, Kohl and Madsen [29] developed an algorithm exploiting Lagrangean relaxation. Then, Kohl, Desrosiers, Madsen, Solomon, and Soumis [30], Larsen [35], and Cook and Rich [7] extended the previous approaches by developing Dantzig-Wolfe based decomposition algorithms involving cutting planes and/or parallel platforms. Kallehauge [25] suggested a hybrid algorithm based on a combination of Lagrangean relaxation and Dantzig-Wolfe decomposition. Recently, Chabrier [5], Chabrier, Danna, and Le Pape [6], Feillet, Dejax, Gendreau, and Gueguen [18], Irnich and Villeneuve [23], and Rousseau, Gendreau, and Pesant [38] have proposed algorithms based on enhanced subproblem methodology. Advancements in master problem approaches have been made by Danna and Le Pape [10] and Larsen [34].

This chapter has the following organization. In section 4.2 we describe the mathematical model of the VRPTW and in section 4.3 we discuss the structure of the problem and decomposition alternatives. Next, sections 4.4 and 4.5 present the master problem and the subproblem for the column generation approach, respectively. Section 4.6 illustrates the branch-and-bound framework, while section 4.7 addresses acceleration strategies used to increase the efficiency of branch-and-price methods. Then, we describe generalizations of the VRPTW in section 4.8 and report computational results for the classic Solomon test sets in section 4.9. Finally we present our conclusions and discuss some open problems in 4.10.

4.2 The model

The VRPTW is defined by a fleet of vehicles, $\mathscr{V}$, a set of customers, $\mathscr{C}$, and a directed graph $\mathscr{G}$. Typically the fleet is considered to be homogeneous, that is, all vehicles are identical. The graph consists of $|\mathscr{C}|+2$ vertices, where the customers are denoted $1,2,\ldots,n$ and the depot is represented by the vertices 0 ("the starting depot") and $n+1$ ("the returning depot"). The set of all vertices, that is, $0,1,\ldots,n+1$ is denoted $\mathscr{N}$. The set of arcs, $\mathscr{A}$, represents direct connections between the depot and the customers and among the customers. There are no arcs ending at vertex 0 or originating from vertex $n+1$. With each arc (i,j), where $i \neq j$, we associate a *cost* c_{ij} and a *time* t_{ij}, which may include service time at customer i.

Each vehicle has a capacity q and each customer i a demand d_i. Each customer i has a *time window* $[a_i,b_i]$ and a vehicle must arrive at the customer before b_i. If it arrives before the time window opens, it has to wait until a_i to service the customer. The time windows for both depots are assumed to be identical to $[a_0,b_0]$ which represents the *scheduling horizon*. The vehicles may not leave the depot before a_0 and must return at the latest at time b_{n+1}.

It is assumed that q,a_i,b_i,d_i,c_{ij} are non-negative integers and t_{ij} are positive integers. Note that this assumption is necessary to develop an algorithm for the shortest path with resource constraints used in the column generation approach presented later. Furthermore it is assumed that the triangle inequality is satisfied for both c_{ij} and t_{ij}.

The model contains two sets of decision variables x and s. For each arc (i,j), where $i \neq j, i \neq n+1, j \neq$

0, and each vehicle k we define x_{ijk} as

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ drives directly from vertex } i \text{ to vertex } j \\ 0, & \text{otherwise} \end{cases}$$

The decision variable s_{ik} is defined for each vertex i and each vehicle k and denotes the time vehicle k starts to service customer i. In case vehicle k does not service customer i, s_{ik} has no meaning and consequently it's value is considered irrelevant. We assume $a_0 = 0$ and therefore $s_{0k} = 0$, for all k.

The goal is to design a set of routes that minimizes total cost, such that

- each customer is serviced exactly once,
- every route originates at vertex 0 and ends at vertex $n+1$, and
- the time windows of the customers and capacity constraints of the vehicles are observed.

This informal VRPTW description can be stated mathematically as a multicommodity network flow problem with time windows and capacity constraints:

$$\text{(4.1)} \qquad \min \sum_{k\in\mathscr{V}} \sum_{i\in\mathscr{N}} \sum_{j\in\mathscr{N}} c_{ij} x_{ijk} \quad s.t.$$

$$\text{(4.2)} \qquad \sum_{k\in\mathscr{V}} \sum_{j\in\mathscr{N}} x_{ijk} = 1 \qquad \forall i \in \mathscr{C}$$

$$\text{(4.3)} \qquad \sum_{i\in\mathscr{C}} d_i \sum_{j\in\mathscr{N}} x_{ijk} \leq q \qquad \forall k \in \mathscr{V}$$

$$\text{(4.4)} \qquad \sum_{j\in\mathscr{N}} x_{0jk} = 1 \qquad \forall k \in \mathscr{V}$$

$$\text{(4.5)} \qquad \sum_{i\in\mathscr{N}} x_{ihk} - \sum_{j\in\mathscr{N}} x_{hjk} = 0 \quad \forall h \in \mathscr{C}, \forall k \in \mathscr{V}$$

$$\text{(4.6)} \qquad \sum_{i\in\mathscr{N}} x_{i,n+1,k} = 1 \qquad \forall k \in \mathscr{V}$$

$$\text{(4.7)} \qquad x_{ijk}(s_{ik} + t_{ij} - s_{jk}) \leq 0 \qquad \forall i, j \in \mathscr{N}, \forall k \in \mathscr{V}$$

$$\text{(4.8)} \qquad a_i \leq s_{ik} \leq b_i \qquad \forall i \in \mathscr{N}, \forall k \in \mathscr{V}$$

$$\text{(4.9)} \qquad x_{ijk} \in \{0,1\} \qquad \forall i, j \in \mathscr{N}, \forall k \in \mathscr{V}$$

The objective function (4.1) minimizes the total travel cost. The constraints (4.2) ensure that each customer is visited exactly once, and (4.3) state that a vehicle can only be loaded up to it's capacity. Next, equations (4.4), (4.5) and (4.6) indicate that each vehicle must leave the depot 0; after a vehicle arrives at a customer it has to leave for another destination; and finally, all vehicles must arrive at the depot $n+1$. The inequalities (4.7) establish the relationship between the vehicle departure time from a customer and its immediate successor. Finally constraints (4.8) affirm that the time windows are observed, and (4.9) are the integrality constraints. Note that an unused vehicle is modeled by driving the "empty" route $(0, n+1)$.

The model can also incorporate a constraint giving an upper bound on the number of vehicles, as is the case in Desrosiers, Dumas, Solomon, and Soumis [14]:

$$\text{(4.10)} \qquad \sum_{k\in\mathscr{V}} \sum_{j\in\mathscr{N}} x_{0jk} \leq |\mathscr{V}| \quad \forall k \in \mathscr{V}, \forall j \in \mathscr{N}$$

Note also that the nonlinear restrictions (4.7) can be linearized as:

$$\text{(4.11)} \qquad s_{ik} + t_{ij} - M_{ij}(1 - x_{ijk}) \leq s_{jk} \quad \forall i, j \in \mathscr{N}, \forall k \in \mathscr{V}$$

The large constants M_{ij} can be decreased to $\max\{b_i + t_{ij} - a_j\}$, $(i,j) \in A$.

For each vehicle, the service start variables impose a unique route direction thereby eliminating any subtours. Hence, the classical VRP subtour elimination constraints become redundant. Finally, the objective function (4.1) has been universally used when solving the VRPTW to optimality. In the research on heuristics it has been common to minimize the number of vehicles which may lead to additional travel cost.

The VRPTW is a generalization of both the traveling salesman problem (TSP) and the VRP. When the time constraints (4.7) and (4.8)) are not binding the problem relaxes to a VRP. This can be modeled by setting $a_i = 0$ and $b_i = M$, where M is a large scalar, for all customers i. If only one vehicle is available the problem becomes a TSP. If several vehicles are available and the cost structure is: $c_{0j} = 1, j \in \mathscr{C}$ and $c_{ij} = 0$, otherwise, we obtain the bin-packing problem. Since trips between customers are "free", the order in which these are visited becomes unimportant and the objective turns to "squeezing" as much demand as possible into as few vehicles (bins) as possible. In case the capacity constraints (4.2) are not binding the problem becomes a m-TSPTW, or, if only one vehicle is available, a TSPTW.

4.3 Structure and decomposition

A closer look at the above model reveals that only the assignment constraints (4.2) are coupling the vehicles while the remaining constraints are dealing with each vehicle separately. This strongly suggests the use of Lagrangean relaxation (LR) or decomposition, for example Dantzig-Wolfe (DWD), to break up the overall problem into a subproblem for each vehicle and a master problem. To date, the most successful decomposition approaches for the VRPTW cast the subproblem as a constrained shortest path structure. The master problem is an integer program whose solution cannot be obtained directly, so its LP relaxation is solved. The column generation process alternates between solving this linear master problem and the subproblem. The former finds new multipliers to send to the latter which uses this information to find new columns to send back. A lower bound on the optimal integer solution of the VRPTW model is obtained at the end of this back and forth process. This is then used within a branch-and-bound framework to obtain the optimal VRPTW solution. If the vehicles are identical, as we have assumed here, all subproblems will be equivalent and therefore it is necessary to only solve one. The master problem and the subproblem will be discussed in more detail in sections 4.4 and 4.5, respectively. The complete column generation process is described in Chapter 1, while the subproblem forms the subject of Chapter 2.

In addition, other LRs are possible but not promising. One may consider relaxing the time and capacity constraints (4.3), (4.7) and (4.8). This yields a linear network flow problem which possesses the integrality property. The corresponding bound can be calculated very fast, but is not likely to be very strong unless capacity is not binding and time windows are very narrow (see Desrosiers, Dumas, Solomon, and Soumis [14]). Relaxing only the capacity or time window constraints also does not seem sensible since the relaxed problem is not generally easier to solve than the original.

Desrochers, Desrosiers, and Solomon [12] were the first to apply DWD with a free number of vehicles. The assignment constraints were considered the coupling constraints, while the subproblem was a shortest path problem with resource constraints. Relaxing the same constraint set and applying LR was first proposed by Kohl and Madsen [29]. Desrosiers, Sauvé, and Soumis [13] have used a similar relaxation to calculate a lower bound for the minimum fleet size for the m-TSPTW.

Jörnsten, Madsen, and Sørensen [24] suggested solving the VRPTW by variable splitting (later called Lagrangean decomposition, or LD). In follow-up work, Halse [20] described three different variable splitting methods where $\sum_j x_{ijk}$ was replaced by y_{ik} in constraint set (4.2) and possibly (4.3). In turn, the constraint $y_{ik} = \sum_j x_{ijk}$ was introduced and Lagrangean relaxed. The problem decomposes into two problems, one in the x- and s-variables and the other in the y-variables. The former problem is further decomposed by vehicle and it is a shortest path problem with resource constraints. The latter is an

assignment-type problem. Specifically, the approaches are:

- **VS1**: Keep constraints (4.2) and (4.3) in the *y*-problem. This represents a generalized assignment problem (GAP) and the *x*/*s*-problem becomes a shortest path problem with time windows (SPPTW). The GAP has the special structure where all right hand sides in (4.3) are identical and d_i does not depend on *k*.
- **VS2**: Keep constraints (4.2) in the *y*-problem. The *y*-problem becomes a "Semi assignment" problem (SAP) consisting of constraints (4.2) only. The *x*/*s*-problem is equivalent to a shortest path problem with time windows and capacity constraints (SPPTWCC). The SAP is easily solvable and possesses the integrality property.
- **VS3**: Keep constraints (4.2) in the y-problem and constraints (4.3) in both the *y*- and the *x*/*s*-problem. The *y*-problem is a GAP and the *x*/*s*-problem constitutes a SPPTWCC.

In the LD master problem, whose role is to find multipliers to the relaxed equation relating *x* and *y*, the number of multipliers is larger than in the LR considered above. This clearly makes the master problem more difficult. Also the subproblems are no longer identical since the LD multipliers depend on both customer and vehicle. Note that only VS1 and VS2 have been implemented.

We now define LB(VS1), LB(VS2) and LB(VS3) as the best lower bounds obtainable from the three variable splitting approaches, respectively. It can be shown that the previous LR and the DWD yield the same lower bound LB(LR/DWD). Provided that the vehicles are identical, Kohl [28] has derived the following results:

$$\begin{aligned} \text{LB(VS3)} &\geq \text{LB(VS1)} \\ \text{LB(VS3)} &\geq \text{LB(VS2)} \\ \text{LB(LR/DWD)} &= \text{LB(VS2)} \end{aligned}$$

There exist instances for which LB(VS3)>LB(VS1). He further showed that LB(VS2)=LB(VS3) under some weak supplementary conditions. This is surprising because it implies there is no additional gain to be derived from solving two hard integer problems (the SPPTWCC and GAP) instead of just one (the SPPTWCC). However, in the more general case where vehicles have different capacities it might be possible that the VS3 model yields a better bound than VS2.

To conclude, in VRPTW case, the variable splitting methods mentioned above generally provide similar lower bounds to those obtained from the ordinary LR or DWD.

4.4 The master problem

The column generation methodology has been successfully applied to the VRPTW by numerous researchers. It represents a generalization of the linear DWD since the master problem and the subproblem are integer and mixed-integer programs, respectively. Often the master problem is simply stated as a set partitioning problem on which column generation is applied, thereby avoiding the description of the DWD on which it is based. To gain an appreciation for different cutting and branching opportunities compatible with column generation, here we present the master problem by going through the steps of the DWD based on the multicommodity network flow formulation (4.1) - (4.9).

The column generation approach exploits the fact that only constraint set (4.2) links the vehicles together. Hence, the integer master problem is defined through (4.1) - (4.2) and (4.9), that is, it contains the objective function, the assignment of customers to exactly one vehicle and the binary requirement on the flow variables. The rest of the constraints and (4.9) are part of the subproblem which has a modified objective function that decomposes into $|\mathcal{V}|$ independent subproblems, one for each vehicle. In the rest

of this section we will focus on the linear master problem (4.1) - (4.2). Branching, necessary to solve the integer master problem, will be discussed in section 4.6.

Let $\mathscr{P}^k$ be the set of feasible paths for vehicle $k, k \in \mathscr{V}$. Hence, $p \in \mathscr{P}^k$ corresponds to an elementary path which can also be described by using the binary values x^k_{ijp}, where $x^k_{ijp} = 1$, if vehicle k goes directly from vertex i to vertex j on path p, and $x^k_{ijp} = 0$, otherwise. Any solution x^k_{ij} to the master problem (4.1) - (4.2) can be written as a non-negative convex combination of a finite number of elementary paths, i.e.,

$$x^k_{ij} = \sum_{p \in \mathscr{P}^k} x^k_{ijp} y^k_p \quad \forall k \in \mathscr{V}, \forall (i,j) \in \mathscr{A} \tag{4.12}$$

$$\sum_{p \in \mathscr{P}^k} y^k_p = 1 \quad \forall k \in \mathscr{V} \tag{4.13}$$

$$y^k_p \geq 0 \quad \forall k \in \mathscr{V}, \forall p \in \mathscr{P}^k \tag{4.14}$$

Using x^k_{ijp} we can define the cost of a path, c^k_p, and the number of times a customer i is visited by vehicle k, a^k_i, as:

$$\begin{array}{rcll} c^k_p & = & \sum_{(i,j) \in \mathscr{A}} c^k_{ij} x^k_{ijp} & \forall k \in \mathscr{V}, \forall p \in \mathscr{P}^k \\ a^k_{ip} & = & \sum_{j \in \mathscr{N} \cup \{n+1\}} x^k_{ijp} & \forall k \in \mathscr{V}, \forall i \in \mathscr{N}, \forall p \in \mathscr{P}^k \end{array}$$

Now we can substitute these values into (4.1) - (4.2) and arrive at the revised formulation of the master problem:

$$\min \sum_{k \in \mathscr{V}} \sum_{p \in \mathscr{P}^k} c^k_p y^k_p \ s.t. \tag{4.15}$$

$$\sum_{k \in \mathscr{V}} \sum_{p \in \mathscr{P}^k} a^k_{ip} y^k_p = 1 \qquad \forall i \in \mathscr{C} \tag{4.16}$$

$$\sum_{p \in \mathscr{P}^k} y^k_p = 1 \qquad \forall k \in \mathscr{V} \tag{4.17}$$

$$y^k_p \geq 0 \quad \forall k \in \mathscr{V}, \forall p \in \mathscr{P}^k \tag{4.18}$$

The mathematical formulation (4.15) - (4.18) is then the linear relaxation of a set partitioning type problem with an additional constraint on the total number of vehicles and a set of convex combination constraints.

In the usual case of a single depot and a homogeneous fleet of vehicles with the same initial conditions for all vehicles, all $\mathscr{P}^k$ are identical, that is, $\mathscr{P}^k = \mathscr{P}, k \in \mathscr{V}$. Furthermore, the networks for the subproblems are also identical. Therefore constraints (4.17) can be aggregated. By letting $y_p = \sum_{k \in \mathscr{V}} y^k_p$, the index k can be eliminated from the formulation (4.15) - (4.18). The resulting model given below is the classical linear relaxation of the set partitioning formulation:

$$\min \sum_{p \in \mathscr{P}} c_p y_p \ s.t. \tag{4.19}$$

$$\sum_{p \in \mathscr{P}} a_{ip} y_p = 1 \quad \forall i \in \mathscr{C} \tag{4.20}$$

$$y_p \geq 0 \quad \forall p \in \mathscr{P} \tag{4.21}$$

In the column generation methodology, the set of columns in the linear master problem is limited to only those that have already been generated, hence the term *restricted* master problem. It consists of finding a set of minimum cost paths among all paths presently in the master problem. The restricted master

problem can mathematically be stated as:

$$\min \sum_{p \in \mathscr{P}'} c_p y_p \ s.t. \tag{4.22}$$

$$\sum_{p \in \mathscr{P}'} a_{ip} y_p = 1 \quad \forall i \in \mathscr{C} \tag{4.23}$$

$$y_p \geq 0 \quad \forall p \in \mathscr{P}' \tag{4.24}$$

Each decision variable y_p counts the number of times path p is used. This is not necessarily integer, but can be any real number in the interval $[0;1]$. The set $\mathscr{P}'$ contains all the paths generated, a_{ip} denotes the number of times customer i is serviced on path p, and, c_p is the cost of the path. The parameter a_{ip} should in principle be either 0 or 1, but since the subproblem is relaxed (see section 4.5) it can take larger integer values.

Solving the restricted master problem yields a solution $y = (y_1, y_2, \ldots, y_{|\mathscr{P}'|})$ which might be integer but this is not guaranteed. If it is integer, a feasible but not necessarily optimal solution to the VRPTW has been found. In addition to the primal solution, a dual solution $\phi = (\phi_1, \phi_2, \ldots, \phi_{|C|})$ is also obtained.

An initial start for the restricted master problem is often the set of routes visiting a single customer, that is, routes of the type depot-*i*-depot (cf. section 4.8). When the optimal solution to the restricted master problem is found, the simplex algorithm asks for a new variable (i.e. a column/path $p \in \mathscr{P} \setminus \mathscr{P}'$) with negative reduced cost. Such a column is found by solving a subproblem, sometimes called the pricing problem. For the VRPTW, the subproblem should solve the problem "Find the path with minimal reduced cost." Solving the subproblem is in fact an implicit enumeration of all feasible paths, and the process terminates when the optimal objective of the subproblem is non-negative (it will actually be 0).

It is not surprising that the behavior of the dual variables plays a pivotal role in the overall performance of the column generation principle for the VRPTW. It has been observed by Kallehauge [25] that dual variables do not converge smoothly to their respective optima. Assume that the paths $(0, i, n+1)$ are used to initialize the algorithm. Figure 4.1 illustrates the instability of the column generation algorithm compared to the stabilized cutting-plane algorithm presented in the above paper. Furthermore, Figure 4.2 illustrates the effect of the size of the multipliers on the computational difficulty of the SPPTWCC subproblems. Whereas the multipliers are large in the Dantzig-Wolfe process, they are small in the cutting-plane approach. This problem originates in the coordination between the master problem and the subproblem.

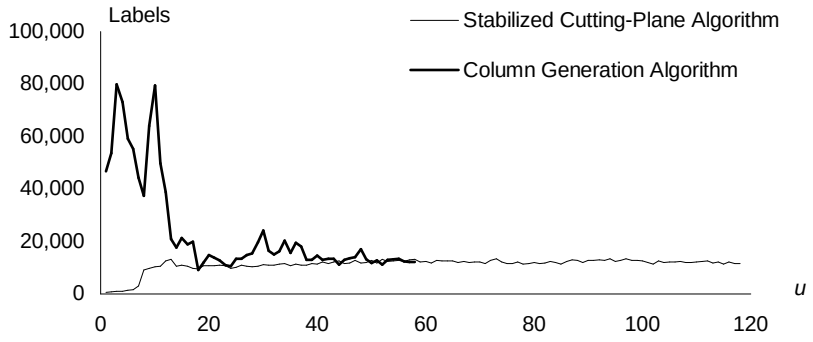
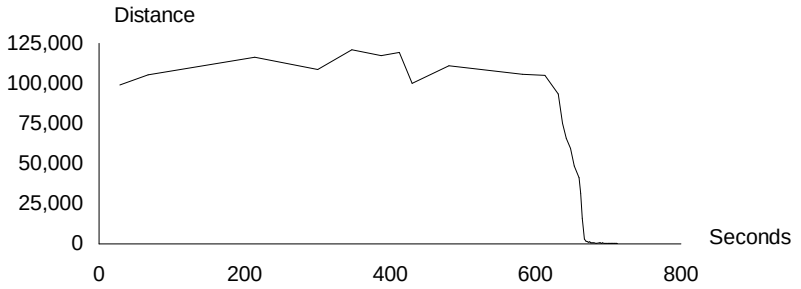
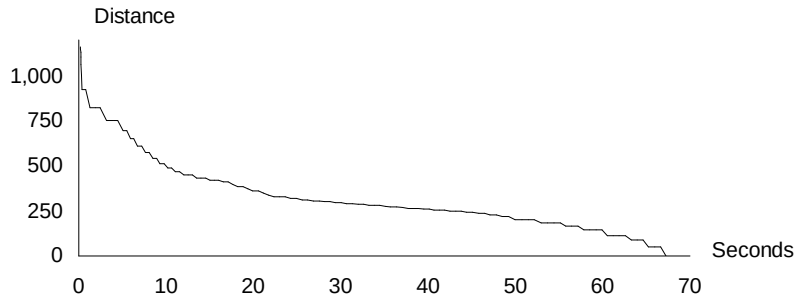


Figure 4.1: Number of labels generated in the subproblem with respect to the iteration number for the Dantzig-Wolfe method and the bundle method on the Solomon instance R104 with 100 customers (from Kallehauge [25]).



(a) Column Generation Algorithm



(b) Stabilized Cutting-Plane Algorithm

Figure 4.2: The Euclidian distance between the current dual variables and the optimum dual variables. Observe the different scales.

This corresponds to the principle of stabilization of column generation as discussed in du Merle, Villeneuve, Desrosiers, and Hansen [16]. Stabilization in the VRPTW context is reported by Kallehauge, Larsen, and Madsen [27]. Here a speedup factor of 6 is reported for the root node of all R1 instances.

Finally, in many routing problems the optimal solution remains unchanged even if overcovering rather than exact covering of customers is allowed. Due to the triangle inequality in the VRPTW, overcovering will always be more expensive than just covering and therefore an optimal solution will always be one where each customer is visited exactly once. The advantage of allowing overcovering is that the linear relaxation of the Set Covering Problem is easier to solve than that of the Set Partitioning Problem, and this will in turn lead to the computation of good estimates of the dual variables.

4.5 The subproblem

In the column generation approach for the VRPTW, the subproblem decomposes into $|\mathcal{V}|$ identical problems, each one being a shortest path problem with resource constraints (time windows and vehicle capacity). More specifically, the subproblem is an Elementary Shortest Path Problem with Time Windows and Capacity Constraints (ESPPTWCC), where elementary means that each customer can appear at most

once in the shortest path. It can be formulated as:

$$\min \sum_{i\in\mathscr{N}} \sum_{j\in\mathscr{N}} \hat{c}_{ij}x_{ij}, \qquad s.t. \tag{4.25}$$

$$\sum_{i\in\mathscr{C}} d_i \sum_{j\in\mathscr{N}} x_{ij} \leq q \tag{4.26}$$

$$\sum_{j\in\mathscr{N}} x_{0j} = 1 \tag{4.27}$$

$$\sum_{i\in\mathscr{N}} x_{ih} - \sum_{j\in\mathscr{N}} x_{hj} = 0 \qquad \forall h \in \mathscr{C} \tag{4.28}$$

$$\sum_{i\in\mathscr{N}} x_{i,n+1} = 1 \tag{4.29}$$

$$s_i + t_{ij} - M_{ij}(1-x_{ij}) \leq s_j \qquad \forall i,j \in \mathscr{N} \tag{4.30}$$

$$a_i \leq s_i \leq b_i \qquad \forall i \in \mathscr{N} \tag{4.31}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \mathscr{N} \tag{4.32}$$

Constraint (4.26) is the capacity constraint, constrains (4.30) and (4.31) are time constraints, while constraint (4.32) ensures integrality. The constraints (4.27), (4.28) and (4.29) are flow constraints resulting in a path from the depot 0 to the depot $n+1$. When solving the ESPPTWCC as the subproblem in the VRPTW, $\hat{c}_{ij}$ is the *modified cost* of using arc (i,j), where $\hat{c}_{ij} = c_{ij} - \pi_i$. Note that while c_{ij} is a non-negative integer, $\hat{c}_{ij}$ can be any real number.

This subproblem does not posses the integrality property, and therefore solving it as a linear mixed-integer programming problem will potentially result in a reduction of the integrality gap between the optimal solution of the LP-relaxed version of the VRPTW and the optimal integer solution to the problem.

Since the ESPPTWCC is NP-hard in the strong sense (see Dror [15] and Kohl [28]), the usual approach has been to slightly alter the problem by relaxing some of the constraints. In particular, allowing cycles changes the problem to the Shortest Path Problem with Time Windows and Capacity Constraints (SPPTWCC). Since arcs can now be used more than once (and customers may therefore be visited more than once), the decision variables x_{ij} and s_i are replaced by x_{ij}^l and s_i^l. The variable x_{ij}^l is set to 1 if the arc (i,j) is used as the l'th arc on the shortest path, and 0 otherwise, and the variable s_i^l is set to the start of service at customer i as customer number l, where $l \in \mathscr{L} = \{1,2,\ldots,|\mathscr{L}|\}$, $|\mathscr{L}| = \lfloor \frac{b_{n+1}}{\min t_{ij}} \rfloor$. The

SPPTWCC can now be described by the following mathematical model:

$$\min \sum_{l\in\mathscr{L}}\sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}} \hat{c}_{ij}x_{ij}^{l}, \qquad s.t. \tag{4.33}$$

$$\sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}} x_{ij}^{1} = 1 \tag{4.34}$$

$$\sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}} x_{ij}^{l} - \sum_{i\in\mathscr{N}}\sum_{j\in\mathscr{N}} x_{ij}^{l-1} \leq 0 \qquad \forall l \in \mathscr{L}-\{1\} \tag{4.35}$$

$$\sum_{i\in\mathscr{C}} d_i \sum_{l\in\mathscr{L}}\sum_{j\in\mathscr{N}} x_{ij}^{l} \leq q \tag{4.36}$$

$$\sum_{j\in\mathscr{N}} x_{0j}^{1} = 1 \tag{4.37}$$

$$\sum_{i\in\mathscr{N}} x_{ih}^{l-1} - \sum_{j\in\mathscr{N}} x_{hj}^{l} = 0 \quad \forall h \in \mathscr{C}\ \forall l \in \mathscr{L}-\{1\} \tag{4.38}$$

$$\sum_{l\in\mathscr{L}}\sum_{i\in\mathscr{N}} x_{i,n+1}^{l} = 1 \tag{4.39}$$

$$s_i^l + t_{ij} - K(1-x_{ij}^l) \leq s_j^l \qquad \forall i,j \in \mathscr{N}\ \forall l \in \mathscr{L}-\{1\} \tag{4.40}$$

$$a_i \leq s_i^l \leq b_i \qquad \forall i \in \mathscr{N} \tag{4.41}$$

$$x_{ij}^l \in \{0,1\} \qquad \forall i,j \in \mathscr{N} \tag{4.42}$$

In this formulation, (4.34) forces the first arc to be used only once, while (4.35) states that arc l can only be used provided that arc $l-1$ is used. The remaining constraints are the original constraints (4.3) to (4.9) extended to include the additional superscript l and the changes related to its inclusion. Note that (4.34) is redundant as it is covered by (4.37), but it has been kept in the model as to indicate the origin node.

This problem can be solved by a pseudo-polynomial algorithm described in Desrochers, Desrosiers, and Solomon [12]. This and all other current approaches are based on dynamic programming. Even though negative cycles are possible, the time windows and the capacity constraints prohibits infinite cycling. Note that capacity is accumulated every time a customer is serviced in a cycle. If the distance used to compute the cost of routes satisfies the triangle inequality, the optimal solution contains only elementary routes. Solving the SPPTWCC instead of the ESPPTWCC augments the size of the set of admissible columns generated for the master problem. Consequently the lower bound on the master problem may decrease. A slight improvement can be obtained by implementing 2-cycle elimination in the solution process which dates back to Kolen, Rinnooy Kan, and Trienekens [31].

While the SPPTWCC relaxation was at the time a computational necessity, the ESPPTWCC has recently been tackled directly. Work on this problem and k-cycle elimination, where $k \geq 3$, proved very successful in expanding the scope of the VRPTW problems solved. Even though the ESPPTWCC continues to be regarded as difficult to solve when time windows are wide, two research groups have recently used it directly in VRPTW optimal algorithms. Chabrier [5] and Chabrier, Danna, and Le Pape [6], and independently Feillet, Dejax, Gendreau, and Gueguen [18] have extended the dynamic programming approach of Desrochers, Desrosiers, and Solomon [12] to the ESPPTWCC by adapting the path dominance rule. They then incorporated several heuristic modifications to make the algorithm much faster. Chabrier [5] and Chabrier, Danna, and Le Pape [6] obtained lower bounds superior to those based on the SPPTWCC resulting in excellent computational results to be described in section 4.9. A different approach that has not yet been tried on the VRPTW is presented in Dumitrescu and Boland [17]. The authors compare three scaling techniques and a standard label-setting method. They show that integrating preprocessing information within the label-setting method can be very beneficial in terms of both memory

and run time. Further improvements of the label-setting method can be obtained by using Lagrangean relaxation.

Instead of dealing with the computational burden of the ESPPTWCC or the weakened lower bound provided by the SPPTWCC, one could consider a middle of the road approach. That is, disallow cycles of small length. As discussed above, cycle elimination corresponding to $k = 2$ has been a common technique. In the SPPTWCC-*k*-cyc, paths with cycles of length of at most *k* are eliminated. The case $k \geq 3$ has been considered by Irnich and Villeneuve [23] with encouraging results presented in section 4.9. Recently Rousseau, Gendreau, and Pesant [38] have presented results where Constraint Programming is used to solve the subproblem. Taking into account the difference in computer power, the authors conclude that their approach is not any faster than that of Desrochers, Desrosiers, and Solomon [12].

4.6 Branch-and-bound

The column generation approach does not automatically guarantee integer solutions and often the solutions obtained will indeed be fractional. Therefore a branch-and-bound framework has to be established. The calculations are organized in a branching tree. For the VRPTW only binary strategies have been proposed in the literature although it should be noted that it is generally not difficult to come up with non-binary branching trees for the problem. The branching decisions are generally based on considerations related to the original 3-index flow formulation (4.2) - (4.9). The column generation process is then repeated at each node in the branch-and-bound tree.

4.6.1 Branching on the number of vehicles

Branching on the number of vehicles was originally proposed by Desrochers, Desrosiers, and Solomon [12]. If the number of vehicles is fractional we introduce a bound on the number of vehicles. Note that this branching strategy does not require that the flow and time variables of the original model be computed.

This branching rule can be implemented fairly easily and only concerns the master problem. We denote the flow over an arc by f_{ij} and this is the sum of all flows over that arc, that is $f_{ij} = \sum_{k \in \mathcal{V}} x_{ijk}$. The f_{ij} values can easily be derived from the solution of the master problem. When we branch on the number of vehicles, two child nodes are created, one imposing on the master problem parent node the additional constraint $\sum_{j \in C} f_{0j} \geq \lceil l \rceil$ while the other forcing $\sum_{j \in C} f_{0j} \leq \lfloor l \rfloor$, where l is the fractional sum of all variables in the master problem.

Note that branching on the number of vehicles is not necessarily enough to obtain an integer solution as it is possible to derive solutions where the sum of the vehicles is integer, but yet there are fractional vehicles driving around the network.

4.6.2 Branching on flow variables

Branching on a single variable x_{ijk} is possible only if each vehicle can be distinguished. In column generation this can be achieved by solving the subproblem for *each* vehicle individually and by introducing an additional constraint in the master problem

$$\sum_{p \in P_k} y_p = 1 \quad \forall k \in \mathcal{V}$$

where P_k is the set of routes generated for each vehicle k and y_p is the binary variable indicating whether route p is used.

Since most cases described in the literature assume a homogeneous fleet, it doesn't make sense to branch on individual vehicles. Instead, branching can be done on sums of flows, that is either on $\sum_j x_{ijk}$

or on $\sum_k x_{ijk}$ (equivalent to f_{ij}). Branching on $\sum_j x_{ijk}$ results in a different subproblem for each vehicle, even though the vehicles are identical. That is because imposing $\sum_j x_{ijk} = 1$ forces customer i to be visited by vehicle k, while $\sum_j x_{ijk} = 0$ implies that customer i is assigned to any vehicle but k.

The standard practice has been to branch on $\sum_k x_{ijk}$ since the branching decision can easily be transferred to the master problem and subproblem. This was proposed independently by Halse [20] and Desrochers, Desrosiers, and Solomon [12]. When $\sum_k x_{ijk} = 1$, customer j succeeds customer i on the same route, while if $\sum_k x_{ijk} = 0$, customer i does not immediately precede j. If there is more than one candidate for branching, that is, there are several fractional variables, we would generally like to choose a candidate that is not close to either 0 or 1 in order to make an impact. When selecting among the nodes to branch on, a common heuristic is to branch on the variable maximizing $c_{ij}(\min\{x_{ijk}, 1 - x_{ijk}\})$ using a best-first strategy In order to create more complex strategies the branching schemes can be applied hierarchically, such as first branching on the number of vehicles and then on $\sum_k x_{ijk}$, or mixed.

4.6.3 Branching on resource windows

Branching on resource windows was first proposed by Gélinas, Desrochers, Desrosiers, and Solomon [19] and is presently the only alternative to branching on flow variables. In the VRPTW model resource windows can be interpreted as either the time windows or the capacity constraints. We will only discuss branching on time windows, as capacity is significantly less constraining in many cases. In Gélinas, Desrochers, Desrosiers, and Solomon [19] only branching on time windows was used.

Branching on time windows results in splitting a time window into two smaller ones. Branching has to be done in such a way that at least one route is infeasible in each of the two sub-windows.

In order to branch on time windows three decisions have to be taken:

1. How should the node for branching be chosen?
2. Which time window should be divided?
3. Where should the partition point be?

In order to decide on the above issues, we define *feasibility intervals* $[l_i^r, u_i^r]$ for all vertices $i \in \mathcal{N}$ and all routes r with fractional flow. l_i^r is the earliest time that service can start at vertex i on route r, and u_i^r is the latest time that service can start, that is, $[l_i^r, u_i^r]$ is the time interval during which route r must visit vertex i to remain feasible.

The intervals can easily be computed by a recursive formula. Additionally we define

$$L_i = \max_{\text{fractional routes } r} \{l_i^r\}, \quad i \in \mathcal{N} \tag{4.43}$$

$$U_i = \min_{\text{fractional routes } r} \{u_i^r\}, \quad i \in \mathcal{N} \tag{4.44}$$

If $L_i > U_i$ at least two routes (or two visits by the same route) have disjoint feasibility intervals, i.e., the vertex is a candidate for branching on time windows. We can branch on a candidate vertex i by dividing the time windows $[a_i, b_i]$ at any integer value in the open interval $[U_i, L_i[$. It should be noted that situations can arise where there are no candidates for branching on time windows, but the solution is not feasible.

Three different strategies were proposed by Gélinas, Desrochers, Desrosiers, and Solomon [19] aiming at the elimination of cycles, the minimization of the number of visits to a customer i and the balancing of flow in the two branch-and-bound nodes.

After having chosen the candidate vertex i for branching, an integer $t \in [U_i, L_i[$ has to be selected in order to determine the division. Here t is chosen in order to divide the time window of the customer such that 1) the flow is balanced and 2) the time window is divided as evenly as possible.

4.7 Acceleration strategies

4.7.1 Preprocessing

The aim of preprocessing is to narrow the solution space by tightening the formulation before the actual optimization is started. This can be done by fixing some variables, reducing the interval of values a variable can take and so on. In the VRPTW, the time windows can be narrowed if the triangle inequality holds. Accordingly, Kontoravdis and Bard [32] propose the following scheme. The earliest time a vehicle can arrive at a customer is by arriving straight from the depot and the latest time it can leave is by going directly back to the depot. Hence, for each customer i, its time window can be strengthened from $[a_i, b_i]$ to $[\max\{a_0 + t_{0i}, a_i\}, \min\{b_{n+1} - t_{i,n+1}, b_i\}]$.

A further reduction of the time windows can be achieved by the method developed by Desrochers, Desrosiers, and Solomon [12]. The time windows are reduced by applying the following four rules in a cyclic manner. The process is stopped when one whole cycle is performed without changing any of the time windows. The four rules are:

1. Minimal arrival time from predecessors:
   $a_l = \max\{a_l, \min\{b_l, \min_{(i,l)}\{a_i + t_{il}\}\}\}$
2. Minimal arrival time to successors:
   $a_l = \max\{a_l, \min\{b_l, \min_{(l,j)}\{a_j - t_{lj}\}\}\}$
3. Maximal departure time from predecessors:
   $b_l = \min\{b_l, \max\{a_l, \max_{(i,l)}\{b_i + t_{il}\}\}\}$
4. Maximal departure time to successors:
   $b_l = \min\{b_l, \max\{a_l, \max_{(l,j)}\{b_j - t_{lj}\}\}\}$

The first rule adjusts the start of the time window to the earliest time a vehicle can arrive coming straight from any possible predecessor. In a similar fashion, the second rule modifies the start of the time window in order to minimize the excess time spent before the time windows of all possible successors open if the vehicle continues to a successor as quickly as possible. The two remaining rules use the same principles to adjust the closing of the time window. With respect to capacity, an arc (i, j) can obviously be removed if $d_i + d_j > q$.

4.7.2 Subproblem strategies

A well known strategy for accelerating column generation is to return many negative marginal cost columns to the master problem. Even though in principle only one needs to be returned, several can be if they are available. Computational tests conducted by Kohl [28] and Larsen [35] confirm the benefits of this approach.

4.7.3 Master problem strategies

Along with the novel perspectives on the subproblem solution described in 4.5, master problem acceleration strategies have been key to the evolution of VRPTW approaches over the last few years. One approach is to accelerate the solution at the root node of the branch-and-bound tree by using a local search method to generate a set of initial columns. This helps the column generation process get a fast increase in the quality of the dual variables. It has been implemented by numerous researchers and has finally been discussed in the literature by Danna and Le Pape [10]. The authors use a local search method based on the savings algorithm incorporating time windows which produces a set of routes better than the trivial

depot-customer-depot ones. Furthermore, local search is used along with a MIP solver throughout the branch-and-price process to generate good integer solutions fast. Two different heuristics, a local search method based on large neighborhood search and a guided tabu search, were tested and proved beneficial, especially on Solomon's R1 and RC1 problem classes.

Two new approaches have been suggested by Larsen [34] and Larsen [35]. First, during the execution of the branch-and-price a large number of columns are generated and many of these only participate in a few computations and will not be used afterwards. If kept, each column will increase computing time when solving the relaxed set partitioning problem and when adjusting the upper bounds on variables due to branching decisions. Therefore Larsen [34] suggests to keep track of how long a column is part of a basis. If it does not participate in a basis for a given number of branch-and-bound nodes it is removed from the model. This was also suggested by Desaulniers, Desrosiers, and Solomon [11] where it was also noted that a certain number of nonbasic columns should remain in the problem. Larsen [34] reports that deleting columns that have not been part of the basis for the last 20 branch-and-bound nodes outperforms the code without column deletion by a factor of 2.5 aggregated over 27 instances.

The second acceleration approach is to stop the algorithm for the SPPTWCC before it completes. Computations can be stopped as soon as at least one route with negative cost has been generated. This approach is denoted "forced early stop" in Larsen [35] and results in dramatic running time reductions, especially for problems with large time windows. For these, the values of the dual variables at the beginning of the procedure will however be of poor quality. Only when the subproblem proves optimality it cannot be stopped prematurely.

4.7.4 Cutting planes

The barebone column generation methodology for solving the VRPTW is part of the popular approach for solving difficult integer programming problems by relaxing the integrality constraints of the original problem. Typically, the optimal solution to the relaxed problem is not feasible for the original problem and branch-and-bound is used in order to get integer solutions.

Cutting planes has been proposed to improve the polyhedral description of the relaxed problem in order to get an integer solution or at least narrow the integrality gap. Kohl, Desrosiers, Madsen, Solomon, and Soumis [30] suggested three cuts in order to tighten the LP formulation of the VRPTW problem. As these cuts are only introduced at the root node, this is not a branch-and-cut approach, where cuts can be introduced at any node of the search tree.

The method is based on subtour elimination constraints and comb inequalities transferred from the TSP, and 2-path cuts. To detect subtour elimination constraints, a separation algorithm by Crowder and Padberg [9] was implemented. With respect to the comb inequalities, only combs with 3 teeth and 2 nodes were detected. The separation algorithm was a primitive enumeration scheme. Neither of these constraints had a large impact on tightening the bound.

A new idea introduced by Kohl, Desrosiers, Madsen, Solomon, and Soumis [30] was the inclusion of 2-path cuts. The basis of this set of cuts is the subtour elimination inequality in the strong form: $x(S) \geq k(S), \forall S \subseteq \mathscr{C}$, where $x(S)$ is the flow leaving the set S, and $k(S)$ is the minimum number of vehicles needed to service the customers in S. Determining $k(S)$ is not an easy task, but using the triangle inequality on the travel times we have that $S_1 \subset S_2 \Rightarrow k(S_1) \leq k(S_2)$. Sets S that satisfy $x(S) < 2$ and $k(S) > 1$ must now be found. As $k(S)$ is an integer, $k(S) > 1$ implies $k(S) \geq 2$. So we need to identify sets S that require at least two vehicles to be serviced, but are currently serviced by less than two.

For a set S, two checks have to be performed: 1) $k(S) > 1$ and 2) can the customers be serviced by a single vehicle? The first check is easy, but the second requires the solution of the TSPTW *feasibility* problem. Since this problem is NP-hard the separation algorithm can only be applied to small sets. This is done heuristically using a simple greedy algorithm based on Laporte, Nobert, and Desrochers [33].

The 2-path cuts outperformed the branch-and-price method without 2-path cuts. The proportion of

the integrality gap closed by the 2-path cuts varies from 100% to 10% in a few cases. Overall 12 new unsolved Solomon instances were closed.

Cook and Rich [7] extended the above 2-path cut approach to k-path cuts involving the solution of a VRPTW with $(k-1)$ customers as part of the separation algorithm. The authors performed experiments with k up to 6. For larger k, the percentage of the integrality gap that is closed is of course larger, but the separation algorithm requires substantially more time and therefore it is not evident that it is preferable to use k larger than 2.

Recently, Bard, Kontoravdis, and Yu [4] have proposed a branch-and-cut algorithm for the arc formulation of the VRPTW. This development parallels the initial uses of this technique for the VRP (Naddef and Rinaldi [37]). Based on the results obtained by Mak [36], a new arc formulation of the VRPTW is presented in Kallehauge and Boland [26]. In this formulation the time and capacity restrictions are modeled using infeasible path elimination constraints (IPECS). This new class of inequalities can be viewed as a strengthening of the IPECS described in Ascheuer, Fischetti, and Grötschel [1], Ascheuer, Fischetti, and Grötschel [2], and Bard, Kontoravdis, and Yu [4] and can also be incorporated at the master problem level in the path formulation considered in this chapter.

Another line of research involves valid inequalities derived from the precedence relationships established by the time windows. That is, if a set of customers is served by the same vehicle, the associated time windows create a precedence structure among the corresponding nodes (Ascheuer, Fischetti, and Grötschel [2]). In Kallehauge and Boland [26], two classes of valid inequalities for the precedence-constrained asymmetric traveling salesman polytope (Balas, Fischetti, and Pulleyblank [3]) are transferred to the VRPTW.

4.8 Generalizations of the VRPTW model

The methods considered in this chapter can be generalized and applied to a number of related problems as discussed by Desrosiers, Dumas, Solomon, and Soumis [14]. Here we will concentrate on routing generalizations and show how a number of more complex routing problems can be modeled based on the framework introduced in the previous sections.

4.8.1 Non-identical vehicles

In the general case vehicles may differ with respect to travel time, travel costs, capacity and possibly other characteristics. We define a class of vehicles as a set of identical vehicles. There may be a cost associated with the vehicles of a particular class, and there may be bounds on their availability as well. These bounds are modeled in to the master problem as supplementary constraints.

The subproblem must be solved separately for each class of vehicles. The marginal costs of the arcs originating at the depot of the subproblem for a particular vehicle class must be modified by the simplex multiplier of the constraints on the availability of this class in the master problem. One can chose to solve one or more of the subproblems between each master iteration. The LP optimality criterion is that no subproblem generates columns with negative reduced costs. It is likely to be efficient to branch on the number of vehicles of a particular class if this number is fractional.

A special case occurs if vehicles do not differ with respect to traveling time, travel cost and time windows, but only have different capacities and possible availability and fixed costs. This problem is clearly solvable as described above, but it can also be transformed into the identical vehicle problem described earlier in this chapter. The advantage of this transformation is that only one subproblem must be solved at each iteration. To illustrate how the transformation works consider a problem with two classes of vehicles, with vehicle capacities q_1 and q_2 respectively, where $q_1 < q_2$. The fixed costs of using the vehicles are c_1 and c_2, respectively. Two extra nodes are inserted in parallel between the depot

and the customers and any path must go through exactly one of these nodes. The two arcs from the depot to the new nodes are priced c_1 and c_2, respectively. If node 1 is chosen, the capacity is reduced by $q_2 - q_1$ since the resource window of node 1 starts at this quantity. Since the resource window of the depot is $[0, q_2]$, a path going through node 1 cannot service customers with accumulated demand of more than $q_2 - (q_2 - q_1) = q_1$. If there are bounds on the availability of the vehicles, these are inserted in the master problem and the simplex multipliers modify the cost of the two new arcs between the depot and the new nodes.

4.8.2 Multiple depots

If the vehicles are based at different depots, one subproblem must be solved for each depot. Constraints on the availability of vehicles at a particular depot are kept in the master problem, and the associated simplex multiplier modifies the cost of arcs originating at the depot. This is equivalent to the general non-identical vehicle case discussed above.

One may assume that the vehicles are allowed to finish their routes at a depot different from the one the vehicles started, but that the number of vehicles starting and ending at any depot remains constant. In this particular case it is sufficient to solve one subproblem. One extra node per depot is created "before" the customers and one "after" the customers. For each depot there will be a constraint r in the master problem requiring the number of vehicles housed at that depot be kept constant. The right hand side will be zero, and the left hand side coefficient (r, p) will be 1 if route p starts at the depot associated with constraint r and ends at another depot, -1 if the route starts at another depot and ends at the depot associated with constraint r, and zero otherwise. The corresponding simplex multipliers modify the cost of arcs originating at the depot (with opposite sign). It is also easy to introduce different fixed costs associated with the vehicles housed at the depots.

4.8.3 Multiple or soft time windows

Customers may have several (disjoint) time intervals in which they can be serviced. A vehicle arriving between two time windows must wait until the beginning of the next time window. This doesn't truly complicate the problem since the usual dominance criterion in the subproblem remains valid. A vehicle arriving at a particular node at time t_1 can do everything a vehicle arriving at time t_2 can, provided that $t_1 < t_2$.

If there exist a cost $c(s_i)$ dependent on the time s_i service at customer i begins, the time window is said to be soft. If the cost is non-decreasing with increasing time this is not problematic, since the dominance criteria remain valid. The most general case where $c(s_i)$ is a general function is not efficiently solvable. Ioachim, Gélinas, Desrosiers, and Soumis [22] present an algorithm for the linear case.

4.9 Computational experiments

Almost from the first computational experiments, a set of problems became the test-bed for both heuristic and exact investigations of the VRPTW. Solomon [39] proposed a set of 168 instances that have remained the leading test set ever since. For the researchers working on heuristic algorithms for the VRPTW a need for bigger problems made Homberger and Gehring [21] propose a series of extended Solomon problems. These larger problems have as many as 1000 customers and several have been solved by exact methods.

4.9.1 The Solomon instances

The test sets reflect several structural factors in vehicle routing and scheduling such as geographical data, number of customers serviced by a single vehicle and the characteristics of the time windows (e.g.,

tightness, positioning and the fraction of time-constrained customers in the instances). Customers are distributed within a $[0,100]^2$ square.

The instances are divided into 6 groups (test-sets) denoted R1, R2, C1, C2, RC1 and RC2. Each of the test sets contain between 8 and 12 instances. In R1 and R2 the geographical data is randomly generated by a random uniform distribution. In the test sets C1 and C2 the customers are placed in clusters, and finally in the RC1 and RC2 test-sets some customers are placed in clusters while others are placed randomly. In the test sets R1, C1 and RC1 the scheduling horizon is short permitting approximately 5 to 10 customers to be serviced on each route. The R2, C2 and RC2 problems have a long scheduling horizon allowing routes with more than 30 customers to be feasible. This makes the problems very hard to solve exactly and they have not been used until recently to test exact methods. The time windows for the test sets C1 and C2 are generated to permit good, maybe even optimal, cluster-by-cluster solutions. For each class of problems the geographical position of the customers is the same in all instances whereas the time windows are changed.

Each instance has 100 customers, but by considering only the first 25 or 50 customers, smaller instances can easily be generated. It should be noted that for the RC-sets this results in the customers being clustered since the clustered customers appear at the beginning of the file. Travel time between two customers is usually assumed to be equal to the travel distance plus the service time at the predecessor customer.

4.9.2 Computational results

This section reviews the results obtained by the best exact algorithms for the VRPTW. All are based on the column generation approach. The tables 4.1 through 4.6 present the solutions for the six different sets of the Solomon instances that have been solved to optimality. Column *K* indicates the number of vehicles used in the optimal solution while the column "Authors" give reference to the first publication(s) of the optimal solution for the problem: Kohl, Desrosiers, Madsen, Solomon, and Soumis [30] (KDMSS), Larsen [35] (L), Kallehauge, Larsen, and Madsen [27] (KLM), Cook and Rich [7] (CR), Irnich and Villeneuve [23] (IV), Chabrier [5] (C), and Danna and Le Pape [10] (DLP). It should be noted that Desrochers, Desrosiers, and Solomon [12] prior to Kohl, Desrosiers, Madsen, Solomon, and Soumis [30] solved 50 of the 87 Solomon problems with narrow time windows, but with different travel times. Whereas all the above mentioned papers compute the travel times using one decimal point precision and truncation, time and cost is computed differently in Desrochers, Desrosiers, and Solomon [12]. Furthermore, solutions to all C1 instances were reported for the first time by Kohl and Madsen [29], who used a Lagrangean relaxation approach.

As discussed in Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis [8], the optimal algorithm of Kohl, Desrosiers, Madsen, Solomon, and Soumis [30] solved 69 of the 87 Solomon benchmark short horizon problems to optimality. Eleven additional problems were solved by Larsen [35], Cook and Rich [7], and Kallehauge, Larsen, and Madsen [27]. Recently, Irnich and Villeneuve [23] were successful in closing three additional instances. Four 100-customer instances are still open.

As also reported in Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis [8], Larsen [35], Cook and Rich [7], and Kallehauge, Larsen, and Madsen [27] also provided exact solutions to 42 of the 81 Solomon long horizon problems. Since then, Irnich and Villeneuve [23], Chabrier [5] and Danna and Le Pape [10] have solved an additional 21 instances, leaving 18 problems still unsolved.

4.10 Conclusions

In this chapter we have highlighted the noteworthy developments for optimal column generation approaches to the VRPTW. To date, such methods incorporating branching and cutting on solutions ob-

Table 4.1: Optimal solutions for the R1 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
R101.25	8	617.1	KDMSS	R107.25	4	424.3	KDMSS
R101.50	12	1044	KDMSS	R107.50	7	711.1	KDMSS
R101.100	20	1637.7	KDMSS	R107.100	11	1064.6	CR+KLM
R102.25	7	547.1	KDMSS	R108.25	4	397.3	KDMSS
R102.50	11	909	KDMSS	R108.50	6	617.7	CR+KLM
R102.100	18	1466.6	KDMSS	R108.100			
R103.25	5	454.6	KDMSS	R109.25	5	441.3	KDMSS
R103.50	9	772.9	KDMSS	R109.50	8	786.8	KDMSS
R103.100	14	1208.7	CR+L	R109.100	13	1146.9	CR+KLM
R104.25	4	416.9	KDMSS	R110.25	5	444.1	KDMSS
R104.50	6	625.4	KDMSS	R110.50	7	697	KDMSS
R104.100	11	971.5	IV	R110.100	12	1068	CR+KLM
R105.25	6	530.5	KDMSS	R111.25	4	428.8	KDMSS
R105.50	9	899.3	KDMSS	R111.50	7	707.2	CR+KLM
R105.100	15	1355.3	KDMSS	R111.100	12	1048.7	CR+KLM
R106.25	5	465.4	KDMSS	R112.25	4	393	KDMSS
R106.50	8	793	KDMSS	R112.50	6	630.2	CR+KLM
R106.100	13	1234.6	CR+KLM	R112.100			

tained through Dantzig-Wolfe decomposition are the best performing algorithms. Valid inequalities have proved an invaluable tool in strengthening the LP relaxation for this class of problems.

Recent advances have stemmed from work on parallel implementations of the overall approach, acceleration strategies, primarily at the master problem level, and the subproblem. Solving the subproblem as a ESPPTWCC or a SPPTWCC-*k*-cyc has shown to be very beneficial. Nevertheless, 25% of Solomon's problems are still unsolved. Additional research in each of these areas should lead to further advances. We expect that the further study of polyhedral structures, parallelism, acceleration strategies, and the subproblem will constitute the backbone of research in this area for the next several years. Master problem acceleration methods relying on local search heuristics is just beginning.

Decomposition algorithms are also easily adaptable to other settings. This is because they comprise modules, such as dynamic programming, that can handle a variety of objectives. Lateness, for one, is becoming an increasingly important benchmark in today's supply chains that emphasize on time deliveries. Moreover, they can be run as optimization-based heuristics by means of early stopping criteria.

We hope that this chapter has shed sufficient light on current developments to lead to exciting further research.

Acknowledgments The research of Marius M. Solomon was partially supported by the Patrick F. and Helen C. Walsh Research Professorship.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36:69–79, 2000. [107]

[2] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem

Table 4.2: Optimal solutions for the C1 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
C101.25	3	191.3	KDMSS	C106.25	3	191.3	KDMSS
C101.50	5	362.4	KDMSS	C106.50	5	362.4	KDMSS
C101.100	10	827.3	KDMSS	C106.100	10	827.3	KDMSS
C102.25	3	190.3	KDMSS	C107.25	3	191.3	KDMSS
C102.50	5	361.4	KDMSS	C107.50	5	362.4	KDMSS
C102.100	10	827.3	KDMSS	C107.100	10	827.3	KDMSS
C103.25	3	190.3	KDMSS	C108.25	3	191.3	KDMSS
C103.50	5	361.4	KDMSS	C108.50	5	362.4	KDMSS
C103.100	10	826.3	KDMSS	C108.100	10	827.3	KDMSS
C104.25	3	186.9	KDMSS	C109.25	3	191.3	KDMSS
C104.50	5	358	KDMSS	C109.50	5	362.4	KDMSS
C104.100	10	822.9	KDMSS	C109.100	10	827.3	KDMSS
C105.25	3	191.3	KDMSS				
C105.50	5	362.4	KDMSS				
C105.100	10	827.3	KDMSS				

Table 4.3: Optimal solutions for the RC1 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
RC101.25	4	461.1	KDMSS	RC105.25	4	411.3	KDMSS
RC101.50	8	944	KDMSS	RC105.50	8	855.3	KDMSS
RC101.100	15	1619.8	KDMSS	RC105.100	15	1513.7	KDMSS
RC102.25	3	351.8	KDMSS	RC106.25	3	345.5	KDMSS
RC102.50	7	822.5	KDMSS	RC106.50	6	723.2	KDMSS
RC102.100	14	1457.4	CR+KLM	RC106.100			
RC103.25	3	332.8	KDMSS	RC107.25	3	298.3	KDMSS
RC103.50	6	710.9	KDMSS	RC107.50	6	642.7	KDMSS
RC103.100	11	1258	CR+KLM	RC107.100	12	1207.8	IV
RC104.25	3	306.6	KDMSS	RC108.25	3	294.5	KDMSS
RC104.50	5	545.8	KDMSS	RC108.50	6	598.1	KDMSS
RC104.100				RC108.100	11	1114.2	IV

with time windows by branch-and-cut. *Mathematical Programming Series A*, 90:475–506, 2001. [107]

[3] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995. [107]

[4] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002. [107]

[5] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. To appear in Computers & Operations Research. [94, 102, 109]

[6] A. Chabrier, E. Danna, and C. Le Pape. Cooperation entre generation de colonnes avec tournees sans cycle et recherche locale appliquee au routage de vehicules. Huitiemes Journees Nationales sur la resolution de Problemes NP-Complets (JNPC 2002). [94, 102]

Table 4.4: Optimal solutions for the R2 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
R201.25	4	463.3	CR+KLM	R207.25	3	361.6	KLM
R201.50	6	791.9	CR+KLM	R207.50			
R201.100	8	1143.2	KLM	R207.100			
R202.25	4	410.5	CR+KLM	R208.25	1	328.2	IV+C
R202.50	5	698.5	CR+KLM	R208.50			
R202.100				R208.100			
R203.25	3	391.4	CR+KLM	R209.25	2	370.7	KLM
R203.50	5	605.3	IV+C	R209.50	4	600.6	IV+C
R203.100				R209.100			
R204.25	2	355	IV+C	R210.25	3	404.6	CR+KLM
R204.50	2	506.4	IV	R210.50	4	645.6	IV+C
R204.100				R210.100			
R205.25	3	393	CR+KLM	R211.25	2	350.9	KLM
R205.50	4	690.1	IV+C	R211.50	3	535.5	IV+DLP
R205.100				R211.100			
R206.25	3	374.4	CR+KLM				
R206.50	4	632.4	IV+C				
R206.100							

[7] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Departement of Computational and Applied Mathematics, Rice University, 1999. [94, 107, 109]

[8] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 157–193. SIAM, Philadelphia, 2002. [109]

[9] H. Crowder and M. Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26:495–509, 1980. [106]

[10] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column generation*, GERAD 25th Anniversary Series, pages 99–129. Springer, New York, 2005. [94, 105, 109]

[11] G. Desaulniers, J. Desrosiers, and M. M. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduliung problems. In C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 309–324. Kluwer Academic Publishers, 2002. [106]

[12] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992. [94, 96, 102, 103, 104, 105, 109]

[13] J. Desrosiers, M. Sauvé, and F. Soumis. Lagrangean relaxation methods for solving the minimum fleet size multiple travelling-salesman problem with time windows. *Management Science*, 34:1005–1022, 1988. [96]

[14] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8

Table 4.5: Optimal solutions for the C2 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
C201.25	2	214.7	CR+L	C205.25	2	214.7	CR+L
C201.50	3	360.2	CR+L	C205.50	3	359.8	CR+KLM
C201.100	3	589.1	CR+KLM	C205.100	3	586.4	CR+KLM
C202.25	2	214.7	CR+L	C206.25	2	214.7	CR+L
C202.50	3	360.2	CR+KLM	C206.50	3	359.8	CR+KLM
C202.100	3	589.1	CR+KLM	C206.100	3	586	CR+KLM
C203.25	2	214.7	CR+L	C207.25	2	214.5	CR+L
C203.50	3	359.8	CR+KLM	C207.50	3	359.6	CR+KLM
C203.100	3	588.7	KLM	C207.100	3	585.8	CR+KLM
C204.25	1	213.1	CR+KLM	C208.25	2	214.5	CR+L
C204.50	2	350.1	KLM	C208.50	2	350.5	CR+KLM
C204.100	3	588.1	IV	C208.100	3	585.8	KLM

Table 4.6: Optimal solutions for the RC2 instances

Problem	*K*	*Dist.*	*Authors*	*Problem*	*K*	*Dist.*	*Authors*
RC201.25	3	360.2	CR+L	RC205.25	3	338	L+KLM
RC201.50	5	684.8	L+KLM	RC205.50	5	630.2	IV+C
RC201.100	9	1261.8	KLM	RC205.100	7	1154	IV+C
RC202.25	3	338	CR+KLM	RC206.25	3	324	KLM
RC202.50	5	613.6	IV+C	RC206.50	5	610	IV+C
RC202.100	8	1092.3	IV+C	RC206.100			
RC203.25	2	326.9	IV+C	RC207.25	3	298.3	KLM
RC203.50	4	490.122	IV+C	RC207.50	4	558.6	C
RC203.100				RC207.100			
RC204.25	3	299.7	C	RC208.25	2	269.1	C
RC204.50	3	444.2	DLP	RC208.50			
RC204.100				RC208.100			

of *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, The Netherlands, 1995. [95, 96, 107]

[15] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994. [101]

[16] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999. [100]

[17] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42:135–153, 2003. [102]

[18] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44:216–229, 2004. [94, 102]

[19] S. Gélinas, M. Desrochers, J. Desrosiers, and M. M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61: 91–109, 1995. [104]

[20] K. Halse. *Modeling and Solving Complex Vehicle Routing Problems*. PhD thesis, Technical University of Denmark, 1992. [94, 96, 104]

[21] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37:297–318, 1999. [108]

[22] I. Ioachim, S. Gélinas, J. Desrosiers, and F. Soumis. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31:193–204, 1998. [108]

[23] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. To appear in INFORMS Journal on Computing. [94, 103, 109]

[24] K. Jörnsten, O. B. G. Madsen, and B. Sørensen. Exact solution of the vehicle routing and scheduling problem with time windows by variable splitting. Technical Report 5/1986, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, 1986. [96]

[25] B. Kallehauge. Lagrangian duality and non-differentiable optimization – applied on vehicle routing. Master's thesis, Technical University of Denmark, 2000. In Danish. [94, 99]

[26] B. Kallehauge and N. Boland. Path inequalities for the vehicle routing problem with time windows. Technical Report 2005-2, Centre for Traffic and Transport, Technical University of Denmark, 2005. [107]

[27] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangean duality and non-differentiable optimization applied on routing with time windows – experimental results. Technical Report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, 2000. [100, 109]

[28] N. Kohl. *Exact Methods for Time Constrained Routing and Related Scheduling Problems*. PhD thesis, Technical University of Denmark, 1995. [97, 101, 105]

[29] N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean relaxation. *Operations Research*, 45:395–406, 1997. [94, 96, 109]

[30] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999. [94, 106, 109]

[31] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987. [102]

[32] G. Kontoravdis and J. F. Bard. A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995. [105]

[33] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1050–1073, 1985. [106]

[34] J. Larsen. Refinements of the column generation process for the vehicle routing problem with time windows. *Journal of Systems Science and Systems Engineering*, 13:326–341, 2004. [94, 106]

[35] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Technical University of Denmark, 1999. [94, 105, 106, 109]

[36] V. H. Mak. *On the Asymmetric Travelling Salesman Problem with Replenishment Arcs*. PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, 2001. [107]

[37] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 49–78. SIAM, Philadelphia, 2002. [107]

[38] L.-M. Rousseau, M. Gendreau, and G. Pesant. Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research*, 130:199–216, 2004. [94, 103]

[39] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–65, 1987. [108]